all rights reversed

HACKER

Γενάρης 2013 # Τιμή τεύχους 6,40€
τεύχος 016 - Lost Winter Edition

Ανάλυση malware, για όλους! [*]



Βάλτε Linux στην κορνίζα (την ψηφιακή)

Παρακολουθήστε αποτελεσματικά το δίκτυό σας

Σχεδιάστε εφαρμογές για το περιβάλλον γραφικών

[*] ...κι αρχίστε να ψαρώνετε άγρια όσους πουλάνε εξυπνάδα ;)

tefhos 16o

Περιεχόμενα

και οδηγίες χρήσης

εντιτόριαλ 6

Όχι και το τόσο τυχαίες σκέψεις, σε όχι και τόσο τυχαία σειρά

σνέιλ μέιλ 8

ο Θείος Ακάκιος προσπαθεί να πείσει τη μητέρα της εξαδέλφης, που είναι Ινδουίστρια, ότι ακόμη κι αν ένα πιθηκάκι πληκτρολογήσει στα σανσκριτικά τις 700 στροφές της Μπάγκαβατ Γκίτα, του έμμετρου διαλόγου μεταξύ του Πρίγκηπα Αρτζούνα και του Κρίσνα λίγο πριν τη μάχη του Κουρουκσέτρα, δεν υπάρχει κανένας λόγος για να ταραχτεί

ανάλυση malware, για όλους! (μέρος 1) 12

πώς μαθαίνουμε όσα περισσότερα μπορούμε για τη συμπεριφορά ενός malware, χωρίς να έχουμε πρόσβαση στο source code

Python, wxPython, wxGlade: Ώρα για Visual Programming! 30

σχεδιάστε εφαρμογές για το περιβάλλον γραφικών (κυριολεκτούμε)

Arduino Sound Master, part 2/όσο χρειαστεί 42

καλούπια για τη μεταβολή της έντασης

52 κυριαρχήστε στο command line part 3/4

αξιώστε πλήρη έλεγχο!

64 ωραιότατη κορνίζα, αλλά για αληθινούς άντρες!

υπάρχουν ψηφιακές κορνίζες στις οποίες μπορεί-τε να βάλετε Linux (και μερικές φορές έχει πολύ πλάκα, για να μη μιλήσουμε για τις εφαρμογές)

74 απόλυτη εποπτεία των μηχανών

έχετε τοπικούς physical servers και VMs ή/και απομακρυσμένα VPSes; αν ναι, χρειαζόσαστε κι έναν τρόπο για να τα παρακολουθείτε!

88 κυριαρχήστε στο command line, part 4/4

προγραμματίστε το BASH

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint A.E. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το http://deltahacker.gr/order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) **πόσο μικρός είναι ο κόσμος; γιατί οι πλούσιοι πλουτίζουν;**
οι απαντήσεις στο **http://deltahacker.gr/?p=7672**

Δαίμονες στο Northern Outpost!

Τον τελευταίο καιρό κάτι έχει αρχίσει ν' αλλάζει, εδώ γύρω. Μη φανταστείτε βέβαια τίποτε εμφανείς αλλαγές. Ακόμα και για εκείνους που μας ξέρουν καλά κι ανά πάσα στιγμή γνωρίζουν τι μας απασχολεί και με τι καταπιανόμαστε, οι αλλαγές στις οποίες αναφερόμαστε είναι εντελώς αόρατες. Και είναι αόρατες επειδή, προς το παρόν, υφίστανται μόνο ως σκέψεις και σχέδια. Αλλά κάπως έτσι δεν ξεκινούν όλες οι μεγάλες, δραματικές αλλαγές;

by subZraw

Έτσι τουλάχιστον λένε στο twitter -- και μάλλον κάπως έτσι πρέπει να είναι. Ίσως το έχετε παρατηρήσει κι εσείς: στο twitter ξέρουν τα πάντα. Σας απασχολούν ζητήματα συναισθηματικής φύσεως; Μήπως οι πολιτικές εξελίξεις, τα επίπεδα της αιθαλομίχλης ή το νέο φορολογικό; Ψάχνετε άραγε το καλύτερο σουβλατζίδικο στη γειτονιά; Θέλετε ίσως να μάθετε πόσα είναι τα στικάκια-αντίγραφα της λίστας Λαγκάρντ; Κανένα πρόβλημα! Μπείτε στο twitter και θα ενημερωθείτε και θα τα μάθετε. Όλα. Και πολλά περισσότερα.

Εγώ πάντως, το έχω ξαναπεί, συχνά πυκνά πιάνω τον εαυτό να ξεχνιέται στο twitter. Μπαίνω για λίγο, έτσι, να δω τι παίζει και να πετάξω καμιά εξυπνάδα ή να τρολάρω τα συνήθη θύματά μου, ξεχνιέμαι όμως και κάθομαι με τις ώρες. Κι αυτό από μόνο του ίσως δεν θα ήταν τόσο κακό, αν δεν παρασυρόμουν κιόλας.

Να, τον τελευταίο καιρό έχω πάθει ζημιά και θέλω ν' αλλάξω τα πάντα, εδώ, στα κεντρικά της Parabing Creations, στο Northern Outpost. Δεν ξέρω αν το γνωρίζετε αλλά η Parabing Creations είναι η μικρή, ανεξάρτητη εκδοτική πίσω από το deltaHacker. Το Northern Outpost είναι το επίσημο ανεπίσημο όνομα των κεντρικών γραφείων της εκδοτικής, εδώ, στη Θεσσαλονίκη (και για την ακρίβεια στην Καλαμαριά). Μεταξύ μας, αυτό το επίθετο, το «κεντρικός», είναι περιττό: Περιφερειακά γραφεία δεν υπάρχουν, εκτός κι αν συνυπολογίσουμε το Southern Outpost του κυρίου αρχισυντάκτη (aka @pvar), έξω από τη Χαλκίδα. Αλλά ξεφεύγω από το θέμα μου.

Θέλω που λέτε να καταγγείλω κάποιους γνωστούς-αγνώστους στο twitter, οι οποίοι διαρκώς με παρασύρουν. Δύο που, καθόλου τυχαία, μου έρχονται τώρα στο μυαλό, είναι οι @giannoug και @sonic2000gr. Αυτοί μάλιστα οι τύποι δεν φτάνει που μας πωρώνουν μέσα από το twitter (και το facebook), μας πωρώνουν και μέσα από τα άρθρα τους στο περιοδικό. Κι εκεί που κάποιοι από εμάς καθόμαστε ήσυχα ήσυχα, χωρίς να πειράζουμε ή να προκαλούμε, να σου και βλέπουμε τα φαινομενικά αθώα τους tweets-slash-posts. Λίγο αργότερα διαβάζουμε και τα άρθρα τους στο περιοδικό. Χωρίς καν να συνειδητοποιήσουμε πώς και γιατί, αρχίζουμε να βλέπουμε τον μικρόκοσμό μας ν' αλλάζει. Κι ενώ οι αλλαγές διαδραματίζονται αρχικά στο μυαλό μας, από πολύ νωρίς έχουμε εκείνη την παράξενη αίσθηση --βεβαιότητα είναι-- ότι σύντομα θα έλθουν τα πάνω κάτω *και* στον φυσικό κόσμο. Ναι, εντάξει, μόνο για τον μικρόκοσμό μας μιλάμε, αλλά αυτό δεν σημαίνει πως για εμάς δεν είναι σημαντικός, ούτε ότι δεν επιδέχεται μεγάλες αλλαγές-slash-βελτιώσεις-slash-προσθήκες.

Καλύτερα τώρα να γίνω λίγο πιο συγκεκριμένος και να σας πω για δύο μόνο αλλαγές-slash-προσθήκες, που θέλω να κάνω στο γραφείο-slash-εργαστήριο της Parabing Creations. Όχι για να με μεταπείσετε να *μην* τις κάνω ή κάτι τέτοιο. Αλλά, να, σκέφτομαι ότι μπορεί να μου έχει ξεφεύγει κάτι και πιστεύω ότι αν είναι έτσι θα μου το θυμίσετε και μετά θα με πείσετε (εύκολα) να το βάλω κι αυτό στον όλο σχεδιασμό.

Ο router που έχω από τον ISP μου, είναι για ανακύκλωση. Από την αρχή ήταν για ανακύκλωση, δηλαδή. Περιττό να πω ότι δεν παίρνει ούτε dd-WRT, ούτε OpenWrt. Αν έπαιρνε, δεν θα ήταν για ανακύκλωση. Αρχικά, λοιπόν, σκεφτόμουν να ψάξω για router που να παίζει

σωστά *τουλάχιστον* με ένα από τα δύο αυτά firmware. Ρώτησα και τον @giannoug σχετικά, ο οποίος έχει πείρα απ' αυτά κι ο άνθρωπος με κατηύθυνε. Έλα όμως που α) ο @sonic2000gr μού έχει κολλήσει εδώ και χρόνια το μικρόβιο του BSD, ενώ ταυτόχρονα β) αγαπώ πολύ το pfSense! Για 'μενα, λοιπόν, η ιδανική λύση για router είναι ένα μικρό, ενεργειακά ευγενικό κι όσο το δυνατόν πιο αθόρυβο box, το οποίο θα τρέχει pfSense. Μετά από το σχετικό ψάξιμο, έχω από τώρα καταλήξει στη μητρική γύρω από την οποία θα στηθεί το σύστημα. Είναι της Jetway κι ένα πλήρες σύστημα που τη χρησιμοποιεί είναι αυτό: http://bit.ly/ibarebone

Ο fileserver έχει πάψει να είναι αξιόπιστος. Τρέχει FreeBSD 9.1-RELEASE και βασίζεται στο ZFS, όμως το hardware έχει αρχίσει να δείχνει την ηλικία του. Αρκεί να πω



ότι όποτε τον ανοίγω, μία στις δύο φορές παίρνω το μήνυμα «no bootable medium found». (Δεν τον αφήνω όλη την ώρα ανοικτό: αφενός δεν χρειάζεται, αφετέρου καλό είναι να κάνουμε και λίγη οικονομία στο ρεύμα.) Στο τοπικό δίκτυο υπάρχει κι ένα PC με triple boot configuration, το οποίο τρέχει Windows 7, Ubuntu 12.04 και Proxmox VE. Το ένα account των Windows 7 είναι αφιερωμένο σε ένα λογιστικό πρόγραμμα, ενώ το άλλο στην παραγωγή του deltaCast (http://deltahacker.gr/category/deltacast). Το Ubuntu πάλι χρειάζεται κι αυτό, αφού όλο και κάτι θέλω να βλέπω και να δοκιμάζω. Το ίδιο ισχύει και για το Proxmox VE.

Ιδανικά, αυτό που θέλω να φτιάξω είναι ένα νέο box, σε ρόλο virtualization host. Το host OS του θα είναι το Proxmox κι ένα από τα guest OSes που πάντα θα τρέχει --για την ακρίβεια θα σηκώνεται αυτόματα, κατά το boot του host-- θα είναι το FreeBSD ή το TrueOS, σε ρόλο fileserver. Εδώ υπάρχει μια σημαντική λεπτομέρεια: Οι δίσκοι του virtualized fileserver δεν θα είναι εικονικοί, αλλά φυσικοί. Κάτι τέτοιο με το KVM είναι εφικτό και μάλιστα ιδιαίτερα χρήσιμο, προφανώς όταν θέλουμε τις μέγιστες δυνατές επιδόσεις. Ο κεντρικός επεξεργαστής του box θα είναι ένας Core i7 (ο πιο χαμηλός σε κατανάλωση), η μνήμη τουλάχιστον 24GB (τα 8GB αφιερωμένα στον ZFS-based fileserver) και, πέρα από έναν SSD για το host και τα guests, θα υπάρχει κι ένας μικρότερος SSD για caching από πλευράς ZFS. Τέλος, παρόντες θα είναι και τουλάχιστον άλλοι τρεις δίσκοι, των 3TB έκαστος και σε RAIDZ configuration.

Σκέτη πώρωση, αγαπητές κι αγαπητοί. Και να φανταστείτε ότι δεν σας είπα για το ιδανικό workstation, το οποίο θα αντικαταστήσει τον γερασμένο (έξι χρόνια είναι αυτά), 24άρη iMac. Σ' αυτό το μηχάνημα θα μεταφερθεί σταδιακά η παραγωγή του deltaCast, ενώ φυσικά θα έχει κι άλλες χρήσεις. Αλλά τη συζήτηση αυτή ας τη συνεχίσουμε άλλη φορά...

Ονέιλ μέιλ



Αγαπητοί μου,

Σας γράφω σήμερα, νωρίς το πρωί πρώτη μέρα του νέου έτους, καθισμένος στην συνηθισμένη μου θέση. Οι άλλοι μέχρι πρότινος ακόμα κοιμόντουσαν, οπότε ήμουν σχετικά απερίσπαστος. Το βλέπω, όμως, ότι η ηρεμία μου έχει ημερομηνία λήξης που πλησιάζει.

Η εξαδέλφη έχει βγει, για να πάρει την εφημερίδα με τους αριθμούς του λαχείου που κερδίζουν.

Φέτος δεν μπορώ να πω, δεν έχω παράπονο με τις γιορτές. Μέχρι στιγμής τα πράγματα εξελίσσονται ομαλά. Και το δέντρο το στολίσαμε παραδοσιακά, και τα κάλαντα τα είπαμε, και τα μελομακάρουνά μας τα φάγαμε, και την βασιλόπιττα την κόψαμε, και τον αφρώδη οίνο Καΐρ τον ήπιαμε, και τα δώρα μας τα ανοίξαμε, και την τριανταμία μας την παίξαμε (για το γούρι), και το λαχείο μας το αγοράσαμε, και γενικώς έχουμε κάνει ό,τι προβλέπεται από τους κανονισμούς.

Μας ήρθε και ο ανηψιός εκ Λονδίνου, και έτσι κάναμε και απαρτία.

Με το που ήρθε, βέβαια, μας έφερε την βροχή και τώρα θέλουμε να τον ξαναστείλουμε πίσω, αλλά, τέλος πάντων, για να μην τα πολυλογώ, καλά έχουμε περάσει.

Το βασικό θέμα συζήτησης αυτές τις μέρες ήταν ο χρόνος, και πόσο γρήγορα περνάει.

Ελαφρώς κοινότυπο, ως θέμα, πράγματι, αλλά τι να κάνουμε, εγώ φταίω που έτσι έχουν τα πράγματα; Κάθε χρονιά που περνάει φαίνεται σαν να περνάει και πιο γρήγορα, και μάλιστα τόσο πιο γρήγορα που έχω αρχίσει να θορυβούμαι.

Δεν ξέρω ποιος είναι υπεύθυνος για το πέρασμα του χρόνου, αλλά μου είναι πλέον προφανές ότι έχει ξεφύγει. Μα εντελώς ξεφύγει. Κάποιος πρέπει να τον πάρει παράμερα για να του μιλήσει. Να του εξηγήσει ότι δεν υπάρχει λόγος βιασύνης. Ο χρόνος περνάει και θα συνεχίσει να περνάει. Ούτε σταματάει, ούτε σκοπεύει να σταματήσει. Ποιος ο λόγος να τον βάλουμε να τρέχει; Ψυχραιμία.

Το άλλο θέμα για το οποίο μιλούσαμε, χτες συγκεκριμένα, είχε να κάνει με την τυχαιότητα και το άπειρο.

Αφορμή αποτέλεσε ένα σχετικώς πρόσφατο κείμενο του Κυρίου Εκδότου περί των μαϊμούδων που γράφουν Σαίξπηρ, ερωτηματικό (http://deltahacker.gr/?p 6608). Μας το θύμισε ο ανηψιός, την ώρα που κόβαμε την πίττα.

Η ιδέα –ή, μάλλον, η νοητική άσκηση– με την πολυγραφότατη μαϊμούδα, είναι απλή. Αρκεί να έχεις άπειρη υπομονή. Κυριολεκτικά. «Αν την αφήσεις να πατάει τυχαία τα πλήκτρα μιας γραφομηχανής», άρχισε να εξηγεί ο ανηψιός, «στο τέλος, δηλαδή μετά από άπειρο χρόνο, αυτή θα σου έχει γράψει ό,τι μπορείς να φανταστείς, από τα άπαντα του Σαίξπηρ, μέχρι το έπος της Μαχαμπαράτα στο πρωτότυπο – ή οδηγούς για την δημιουργία undetectable backdoors».

Η Μαιρούλα πήγε να ρωτήσει τι είναι τα undetectable backdoors, αλλά δεν το τόλμησε.

Η μητέρα της Εξαδέλφης, που είναι Ινδουίστρια, παρέμενε σιωπηλή. Ήταν φανερό πως η εικόνα της μαϊμούς που πληκτρολογεί στα σανσκριτικά τις 700 στροφές της Μπάγκαβατ Γκίτα, του έμμετρου διαλόγου μεταξύ του Πρίγκηπα Αρτζούνα και του Κρίσνα λίγο πριν τη μάχη του Κουρουκσέτρα, την είχε ταράξει.

Εγώ παρέμενα σιωπηλός.

«Δηλαδή τι εννοείς, 'ό,τι μπορείς να φανταστείς';» ρώτησε η Κυρία Καίτη. «Πώς το αποφασίζει η μαϊμού τι από όλα θα γράψει;»

«Η μαϊμού δεν αποφασίζει τίποτα. Στην τύχη πατάει τα πλήκτρα. Απλά οι νόμοι των πιθανοτήτων μάς υπαγορεύουν το συμπέρασμα ότι, στο τέλος, θα έχει γράψει τα πάντα: _και_ τα άπαντα του Σαίξπηρ _και_ ό,τι άλλο βάλει ο νους μας. Ή, για να το πούμε αλλιώς, η πιθανότητα να *μην* τα έχει γράψει ισούται με μηδέν.»

Μείναμε πάλι σιωπηλοί.

«Ναι», λέει η Κυρία Καίτη. «Αλλά αν είναι έτσι, είναι σαν να μου λες ότι η μαϊμού όχι μόνο θα έχει γράψει όλον τον Σαίξπηρ, αλλά και όλες τις μεταφράσεις του Σαίξπηρ, σε όλες τις γλώσσες, γνωστές και άγνωστες, όλα τα έργα όλων των άλλων 'Σαίξπηρ' του παρελθόντος, ή των 'Σαίξπηρ' του μέλλοντος, και πάει λέγοντας. Αυτό λες;»

«Ναι, αυτό λέω», ήταν η απάντηση του ανηψιού.

«Μα τότε, τι νόημα έχει να παιδευόμαστε; Για ποιον λόγο να κάτσει ένας υποψήφιος 'Σαίξπηρ' του μέλλοντος να τελειοποιήσει το ταλέντο του, αφού η μαϊμού θα τον έχει προλάβει;»

«'Θα τον έχει προλάβει' είναι σχήμα λόγου, βέβαια, καθώς η μαϊμού θα χρειαστεί άπειρο χρόνο...»

«Και πάλι, όμως. Για ποιο λόγο να ασχοληθεί, αν από μαθηματικής πλευράς το έργο του -έστω κι αν ακόμα είναι υποθετικό και, ουσιαστικά, ανύπαρκτο- εμπεριέχεται στο έργο της μαϊμούς;»

Τα ερωτηματικά της Κυρίας Καίτης άγγιζαν, κατά τη γνώμη μου, την καρδιά, την φιλοσοφική καρδιά, του ζητήματος, και αυτό με παρακίνησε να μπω κι εγώ στη συζήτηση.

Υπάρχει, προσπάθησα να εξηγήσω, μια θεμελιώδης διαφορά μεταξύ της μαϊμούδας και του Σαίξπηρ, και αυτή δεν βρίσκεται στο ότι η μαϊμού έχει άπειρο χρόνο στη διάθεσή της, ούτε στο ότι πατάει τυχαία τα πλήκτρα μιας γραφομηχανής ενώ ο Σαίξπηρ (ο ορίτζιναλ Σαίξπηρ) έγραφε με πένα και μελάνι.



Η βασική τους διαφορά είναι ότι ο Σαίξπηρ *ήθελε* να γράψει τα έργα του. Η απόφαση να τα γράψει ήταν δική του, ενώ η μαϊμού του νοητικού μας πειράματος πατάει πλήκτρα επειδή εμείς την έχουμε βάλει να τα πατάει. Αν ήταν στο χέρι της, θα παρατούσε τη γραφομηχανή και θα ανέβαινε στο δέντρο να βρει τις φιλενάδες της.

Θα μου πείτε ότι αυτό που σχολιάζω είναι αυτονόητο, αλλά αν το σκεφτείτε προσεκτικά θα δείτε ότι δεν είναι.

Το νοητικό πείραμα της πολυγραφότατης μαϊμούς μας καλεί να βάλουμε προσωρινά στην άκρη την διαφορά αυτή, με βάση τη διαπίστωση ότι, από μαθηματικής πλευράς, δεν είναι σημαντική. Αυτό συσκοτίζει το ζήτημα. Το ζήτημα συσκοτίζεται ακόμα περισσότερο με το ανακριβές συμπέρασμα ότι στο τέλος η μαϊμού θα έχει γράψει τα άπαντα του Σαίξπηρ ή την Μπάγκαβατ Γκίτα στα σανσκριτικά.

Το λέω ανακριβές επειδή, strictly speaking, αυτό που θα έχει γράψει η μαϊμού θα είναι μια συμβολοσειρά άπειρου μήκους, σε σημεία της οποίας θα εμφανίζονται τμηματικές συμβολοσειρές λιγότερο ή περισσότερο όμοιες με αυτές που αντιπροσωπεύουν τα έργα του Σαίξπηρ, την Μπάγκαβατ Γκίτα ή τους οδηγούς δημιουργίας undetectable backdoors.

Το λάθος στο νοητικό μας πείραμα, το συλλογιστικό του πρόβλημα, είναι πως επιτρέπουμε να μετατραπεί η μορφολογική σύγκριση δύο οντοτήτων σε σύγκριση των διαδικασιών παραγωγής τους. Λέμε, δηλαδή, πως η μαϊμού έχει γράψει Σαίξπηρ, ενώ το μόνο που έχει αυτή γράψει είναι μια τυχαία συμβολοσειρά (string) απείρου μήκους. Το ότι τμήματα της συμβολοσειράς (substrings) μπορεί να βγάζουν true αν συγκριθούν με Σαίξπηρ ή δεν ξέρω τι άλλο, είναι αδιάφορο.

Σας θυμίζει τίποτα αυτό το συλλογιστικό πρόβλημα;

Θα θυμάστε, ίσως, που σας έγραφα στην περασμένη μου επιστολή για το πώς οι επιστήμονες της ψυχολογίας «συγχέουν τις ερευνητικές τους υποθέσεις με τις ερευνητικές τους παρατηρήσεις και μετά από λίγο ξεχνάνε ποιο ήταν ποιο».

Ε, λοιπόν, το ίδιο πρόβλημα έχουμε και εδώ. Κατά τον ίδιο τρόπο που στην περίπτωση της ψυχοπαθολογίας η περιγραφή ενός φαινομένου εκλαμβάνεται ως αυτοτελής οντότητα, στην περίπτωσή μας εδώ η μορφική ομοιότητα δύο προϊόντων (δηλαδή η «περιγραφή» τους) εκλαμβάνεται ως αποτέλεσμα «όμοιων» μηχανισμών.

Για να το συνδέσω και με αυτά που έλεγα περί νοημοσύνης, το ίδιο πρόβλημα έχουμε και όταν μπερδεύουμε την αυτοαναφορική περιγραφή της συμπεριφοράς ενός συστήματος με τις υποθέσεις μας περί των ιδιοτήτων ή των χαρακτηριστικών του συστήματος αυτού.

Θα το πω επιγραμματικά: Το μόνο που καθιστά την τάδε ή τη δείνα συμβολοσειρά έργο του Σαίξπηρ, απόσπασμα της Μπάγκαβατ Γκίτα ή οδηγό για undetectable backdoors, είναι το ότι πρόκειται για δημιούργημα της θέλησης (ή επιθυμίας, ή ανάγκης) ενός ανθρωπίνου όντος, το οποίο (δημιούργημα) αναγνωρίζεται από τα άλλα ανθρώπινα όντα ως τέτοιο, δηλαδή ως ανθρώπινο δημιούργημα.

Για τον ίδιο ακριβώς λόγο ένα όμορφο ηλιοβασίλεμα, όσο όμορφο κι να είναι, δεν είναι *ποτέ* έργο τέχνης.

Και αντιστρόφως.

Εδώ όμως θα πρέπει να σας αφήσω. Επέστρεψε η εξαδέλφη με την εφημερίδα.

Υπάρχουν προτεραιότητες, καταλαβαίνετε.

Σας ασπάζομαι,
Θείος Ακάκιος

ονέλ μέλ

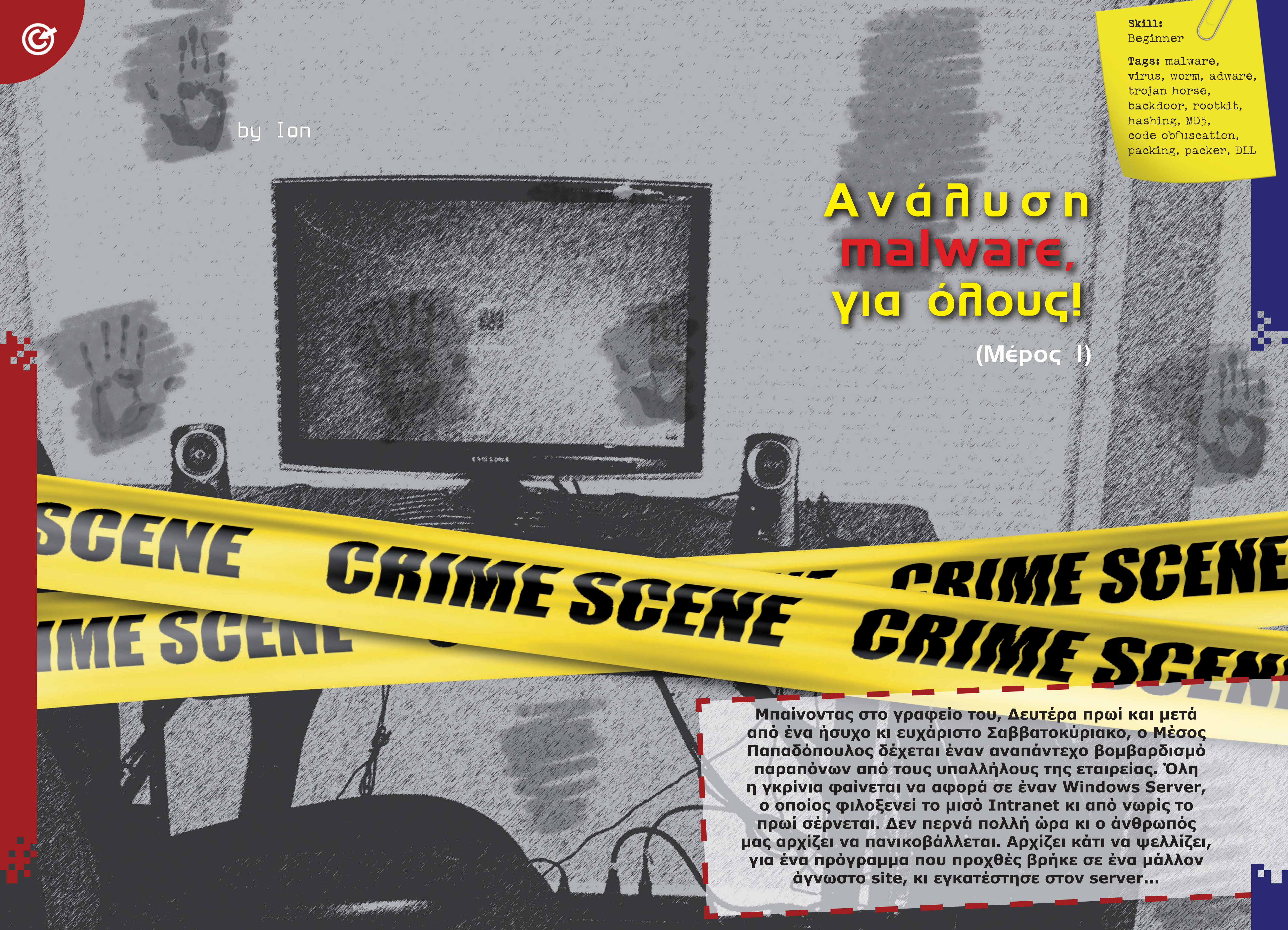
Skill: Beginner

Tags: malware, virus, worm, adware, trojan horse, backdoor, rootkit, hashing, MD5, code obfuscation, packing, packer, DLL

by Ion

Ανάλυση malware, για όλους!

(Μέρος I)



Μπαίνοντας στο γραφείο του, Δευτέρα πρωί και μετά από ένα ήσυχο κι ευχάριστο Σαββατοκύριακο, ο Μέσος Παπαδόπουλος δέχεται έναν αναπάντεχο βομβαρδισμό παραπόνων από τους υπαλλήλους της εταιρείας. Όλη η γκρίνια φαίνεται να αφορά σε έναν Windows Server, ο οποίος φιλοξενεί το μισό Intranet κι από νωρίς το πρωί σέρνεται. Δεν περνά πολλή ώρα κι ο άνθρωπός μας αρχίζει να πανικοβάλλεται. Αρχίζει κάτι να ψελλίζει, για ένα πρόγραμμα που προχθές βρήκε σε ένα μάλλον άγνωστο site, κι εγκατέστησε στον server...

Το συγκεκριμένο εργαλείο υποτίθεται ότι έπαιρνε αυτοματοποιημένα backup ιστοσελίδων, κι ο **Μ**έσος **Π**αναδόπουλος το βρήκε χρήσιμο για ένα πρότζεκτ που τώρα τελευταία δουλεύει. Όταν όμως το έτρεξε, το πρόγραμμα δεν φάνηκε να κάνει κάτι. Ντάμπλγιου-τι-εφ, μουρμούρισε ο Μ.Π. κι ελαφρά εκνευρισμένος άρχισε την αναζήτηση για κάποιο άλλο utility. Ήταν Παρασκευή απόγευμα, η ώρα για να σχολάσει πλησίαζε κι από πάνω είχε και το τριβέλισμα δύο και πλέον εβδομάδων από άλλους συναδέλφους, οι οποίοι τον ζάλιζαν εξυμνώντας τα οφέλη των αυτοματισμών, των αντιγράφων εφεδρείας κ.λπ. κ.λπ.

Παίρνοντας μια βαθιά ανάσα, ο Μ.Π. συνδέεται τώρα στον Windows Server και ψάχνει να βρει τι μπορεί να πηγαίνει στραβά. Ήδη βέβαια διάφορες σκέψεις περνούν από το μυαλό του, αλλά μην τολμώντας καν να τους δώσει την πρέπουσα βαρύτητα ελπίζει ότι το πρόβλημα θα οφείλεται Κάπου Αλλού (TM). Μετά από λίγη ώρα απρόθυμα επιστρέφει στην αρχική, φρικτή υποψία, που πλέον έχει γίνει βεβαιότητα.

Με εκτενές ψάξιμο παρατηρεί κάποια αρχεία τα οποία δεν ήταν εκεί την Παρασκευή, ενώ ούτε και σήμερα θα έπρεπε να είναι. Η επισκόπηση των ενεργών διεργασιών του συστήματος αρχικά δεν δείχνει κάτι το ύποπτο. Ξαφνικά, όμως, παρατηρεί δυο εφαρμογές που συνήθως είναι idle, τώρα να είναι κάπως υπερδραστήριες. Επιπρόσθετα, έχουν δεσμεύσει και ύποπτα πολλή μνήμη! Με μια γρήγορη ματιά στα logs του firewall αποδεικνύεται πως έχουν και διάθεση για κουβέντα, καθώς πολλά πακέτα φαίνεται να πηγαινοέρχονται εξαιτίας τους σε non-standard ports!

Οι κακές σκέψεις που έχουν κατακλύσει τον νου του Μ.Π. έχουν πλεον επιβεβαιωθεί. Απρόθυμα και κάπως φοβισμένα φωνάζει τον junior admin για βοήθεια, ενώ ενημερώνει και τον προϊστάμενο για μια κακόβουλη εφαρμογή, η οποία έχει εγκατασταθεί στον Windows Server και πιθανώς και σε άλλα μηχανήματα της εταιρείας. *«Το πώς δεν έχει σημασία»*, προσθέτει, προσπαθώντας να προσδώσει στον τόνο της φωνής του την αίσθηση του κατεπείγοντος. *«Τώρα είναι η ώρα της δράσης — και πρέπει να δράσουμε χθες!»*, συμπληρώνει, ελπίζοντας ότι αυτό το τελευταίο θα μουδιάσει αρκετά τον προϊστάμενο κι έτσι θα κερδίσει χρόνο και κυρίως μια κάποια άνεση κινήσεων, προκειμένου να δει τι στο καλό θα κάνει. Σε κάθε περίπτωση, έχει μια μεγάλη μέρα μπροστά του.

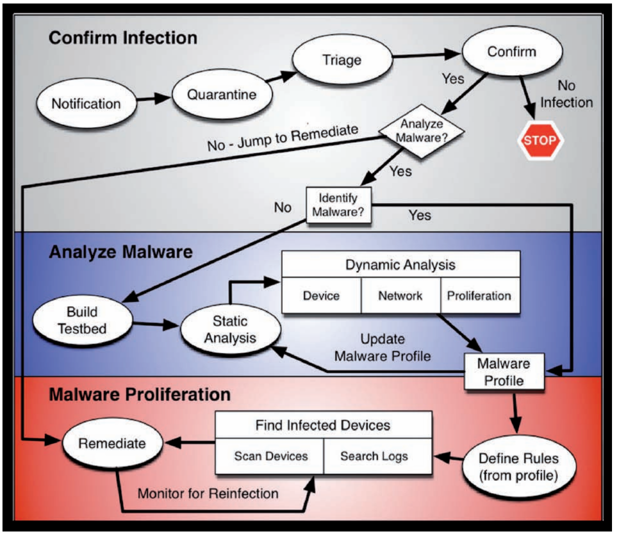
Τι διαγράφεται (μπροστά μας)

Αγαπητές φίλες και φίλοι, στη σειρά άρθρων που τώρα ξεκινά θα μάθουμε πώς μπορούμε να αναλύουμε κακόβουλα προγράμματα, σαν αυτό που θρονιάστηκε στον υπολογιστή του Μέσου Παπαδόπουλου. Κι όταν μιλάμε για ανάλυση εννοούμε την εφαρμογή διαφόρων τεχνικών, ώστε να μαθαίνουμε τις ενέργειες που εκτελεί ένα μεταγλωττισμένο αρχείο (binary) για το οποίο, φυσικά, δεν διαθέτουμε τον πηγαίο κώδικα (source code). Πρακτικά, έτσι έχουν πάντα τα πράγματα με τους ιούς, τα worms, τα trojan horses κι άλλα κακόβουλα προγράμματα (malware).

Με τις τεχνικές μας θα καταλαβαίνουμε τι σκοπό έχει το malware, ποια ήταν τα κίνητρα του δημιουργού του, πώς είναι προστατευμένο, πώς κρύβεται από τα μάτια του μέσου χρήστη κ.ο.κ. Καλό είναι να γνωρίζετε ότι η διαδικασία της ανάλυσης γίνεται λίγο πολύ στα τυφλά — τουλάχιστον στα πρώτα στάδιά της. Βεβαίως, καθένα από τα στάδια ενδέχεται να ανατροφοδοτεί με νέες πληροφορίες γνώσεις που εξήχθησαν σε προηγούμενα. Κατά τον τρόπο αυτό, είναι προφανές ότι συχνά πραγματοποιείται ένας κύκλος απόκτησης πληροφοριών.

Θα ασχοληθούμε με δύο βασικές τεχνικές ανάλυσης, ενώ θα παρουσιάσουμε και τα εργαλεία που χρησιμοποιούνται σε κάθε μία. Φυσικά, επειδή μας αρέσουν οι πράξεις περισσότερο από τα λόγια, θα εφαρμόσουμε όλα αυτά πάνω σε πολύ συγκεκριμένα δείγματα κακόβουλων προγραμμάτων.

Στο τέλος της σειράς μας θα έχετε αποκτήσει την ικανότητα διενέργειας μιας βασικής ανάλυσης ενός ύποπτου προγράμματος, ενώ θα μπορείτε να βγάζετε και συμπεράσματα για το τι ενέργειες θα πραγματοποιήσει σε περίπτωση που ενεργοποιηθεί. Πριν περάσουμε όμως στην ίδια τη διαδικασία της ανάλυσης, ας βάλουμε μια σειρά ξεκινώντας από τα βασικά.



Ποιος είπε ότι η αντιμετώπιση κακόβουλου λογισμικού είναι μια απλή, ακολουθιακή διαδικασία; Κάθε άλλο: Υπάρχουν πολλά βήματα που εκτελούνται επαναληπτικά, ειδικά για το κομμάτι της ανάλυσής του!

Κακόβουλο λογισμικό

Το κακόβουλο λογισμικό ή malware (από το malicious software), είναι ένα από τα πιο παλιά προβλήματα στον τομέα της ασφάλειας υπολογιστών και δικτύων. Από τον πρώτο ιό για PC που εμφανίστηκε το 1986 ως και τις ημέρες μας, το ζήτημα του malware έχει πάρει τεράστιες διαστάσεις. Κάποτε τα κακόβουλα προγράμματα γράφονταν για πλάκα ή έστω έκαναν την εκτέλεσή τους προφανέστατη. Σήμερα έχουν φτάσει σε σημείο να χρησιμοποιούνται ως κρυφά επιθετικά όπλα, με γεωπολιτικά κίνητρα πίσω από την κατασκευή τους. (Δείτε π.χ. τις περιπτώσεις του Stuxnet ή του Flame και θα καταλάβετε σε τι αναφερόμαστε.)

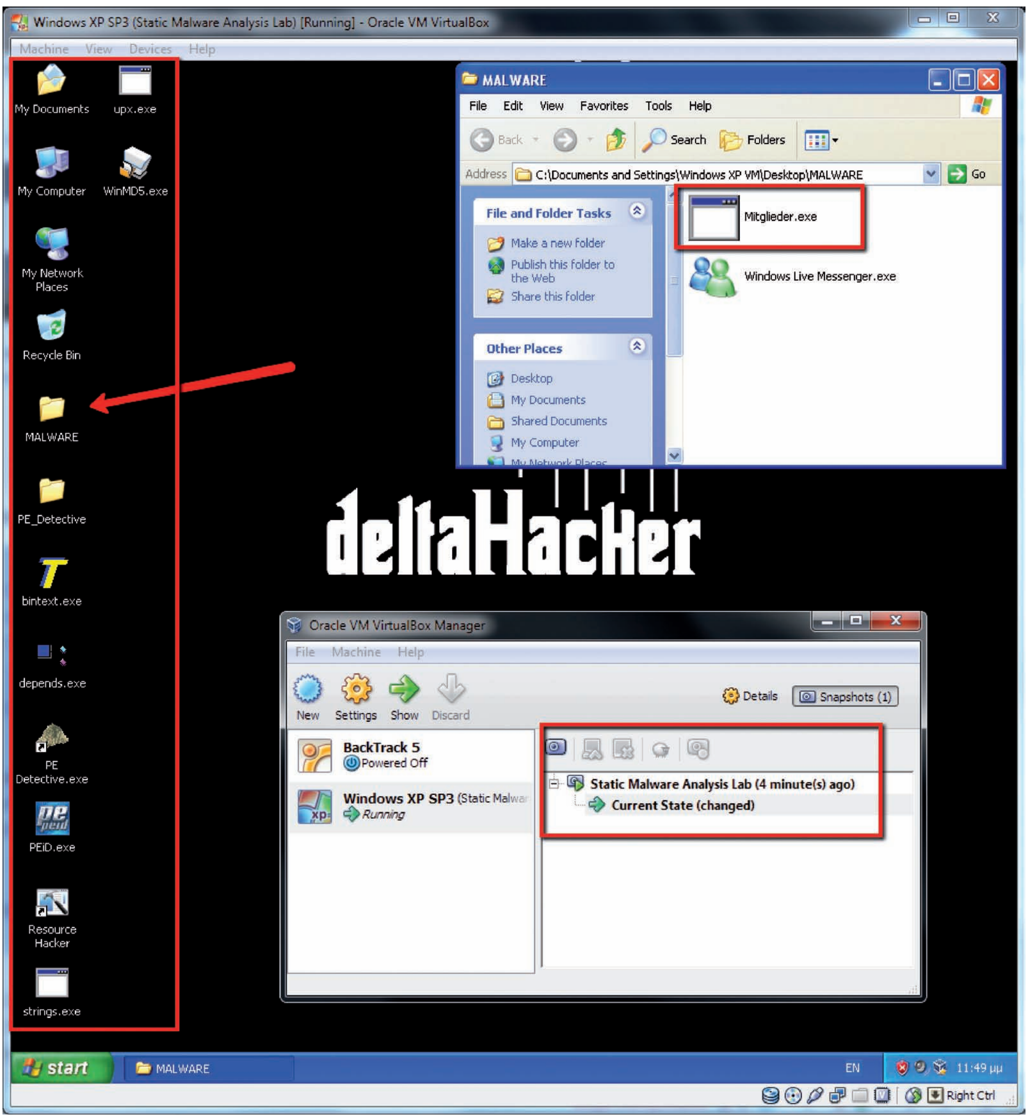
Γενικά, το κακόβουλο λογισμικό έχει σχεδιαστεί προκειμένου να προκαλεί διακοπή ή άρνηση κάποιας υπηρεσίας, να συγκεντρώνει (παράνομα) πληροφορίες και ιδιωτικά/προσωπικά στοιχεία χρηστών, να επιτρέπει την απομακρυσμένη διαχείριση των μολυσμένων μηχανημάτων κ.λπ. Τα επίσημα συμπτώματα από την μόλυνση ενός υπολογιστή με κακόβουλο λογισμικό, ενδέχεται να είναι ένα ή περισσότερα από τα ακόλουθα.

- Καθυστέρηση στη λειτουργία (το μηχάνημα σέρνεται). Αυτό συμβαίνει καθώς το malware χρησιμοποιεί πολλούς από τους διαθέσιμους πόρους του συστήματος.
- Εμφάνιση νέων, άγνωστων εκτελέσιμων αρχείων στο σύστημα. Αυτά συνήθως παράγονται από το ίδιο το κακόβουλο αρχείο μετά την εκτέλεση του — ή λαμβάνονται αυτόματα από το Διαδίκτυο.
- Ανεπιθύμητη δικτυακή δραστηριότητα, όπως, π.χ., η μεταφορά πακέτων από και προς άγνωστες διευθύνσεις IP ή/και ιστοσελίδες, τις οποίες ο χρήστης δεν επισκέπτεται.
- Αλλαγές σε ρυθμίσεις του συστήματος ή εγκατεστημένων προγραμμάτων, όπως, π.χ., η αρχική σελίδα του web browser.
- Εμφάνιση παραθύρων pop-up και διαφημίσεων που πριν το malware δεν εμφανίζονταν.

Τώρα, για να μπορέσουμε να χαρακτηρίσουμε ένα πρόγραμμα ως κακόβουλο, μ’ άλλα λόγια ως malware, θα πρέπει να εκτελεί μία ή περισσότερες από τις ακόλουθες λειτουργίες.

- Αλλαγή των ρυθμίσεων ενός άλλου προγράμματος.
- Αναπαραγωγή του εαυτού του τοπικά ή/και αποστολή του εαυτού του σε άλλους υπολογιστές στο δίκτυο, εννοείται κρυφά και μυστικά.
- Παροχή απομακρυσμένης πρόσβασης στο σύστημα όπου βρίσκεται, σε κάποιον επιτιθέμενο.
- Συγκέντρωση προσωπικών στοιχείων ή ευαίσθητων πληροφοριών

Το εργαστήριο μας είναι έτοιμο! Η εικονική μηχανή φορτώθηκε με τα κατάλληλα προγράμματα αλλά κι ένα trojan horse, έτοιμο ν’ αναλυθεί!



(κωδικοί πρόσβασης, αριθμοί πιστωτικών καρτών κ.λπ.) και αποστολή τους σε απομακρυσμένη τοποθεσία.

- Άνοιγμα ενός ή περισσοτέρων ports στο σύστημα κι αναμονή για συνδέσεις σε αυτά, ενδεχομένως για τη λήψη εντολών περαιτέρω δράσης.
- Καταγραφή όσων πληκτρολογούνται στο σύστημα κι αποστολή των πληροφοριών σε απομακρυσμένο, κακόβουλο χρήστη.
- Σύνδεση σε απομακρυσμένους υπολογιστές, εννοείται χωρίς άδεια.
- Αυτόματη λήψη κι εκτέλεση αρχείων από το Διαδίκτυο.
- Εισαγωγή κώδικα (code injection) σε άλλα προγράμματα ή/και διεργασίες.
- Τροποποίηση των ρυθμίσεων ασφαλείας του συστήματος.
- Αλλαγή των παραμέτρων του registry σε συστήματα Windows. Οι αλλαγές συνήθως αφορούν στην αυτόματη εκτέλεση προγραμμάτων κατά την εκκίνηση του λειτουργικού.

Αφού καθορίσαμε σε υψηλό επίπεδο τη δράση και τις συνέπειες των malware, ας δούμε ποιες ακριβώς είναι και οι κατηγορίες αυτών των προγραμμάτων. Ο μέσος χρήστης αποκαλεί «ιό» ή «virus» οτιδήποτε θυμίζει κακόβουλο λογισμικό, στην πραγματικότητα όμως ένας ιός είναι απλά μία μόνο κατηγορία κακόβουλου λογισμικού, από τις πολλές που υπάρχουν.

Ιός (Virus). Πρόγραμμα που είναι συνήθως ενσωματωμένο (κρυμμένο) μέσα σε άλλο (φαινομενικά) αθώο λογισμικό. Παράγει αντίγραφα του εαυτού του και τα εισάγει μέσα σε άλλα προγράμματα και συνήθως εκτελεί κάποια κακόβουλη ενέργεια, όπως, π.χ., είναι η διαγραφή κρίσιμων αρχείων. Ο τρόπος εξάπλωσής του μοιάζει με αυτόν μιας ασθένειας κι από εκεί παίρνει το όνομά του. Θα μπορούσε να είναι ένα εκτελέσιμο αρχείο ή ένα script.

Σκουλήκι (Worm). Είναι συνήθως ένα μικρό, αυτόνομο πρόγραμμα το οποίο έχει την ικανότητα να αυτό-αναπαράγεται και να μεταφέρεται από υπολογιστή σε υπολογιστή μέσα σ’ ένα δίκτυο. Παράλληλα, εκτελεί κάποια κακόβουλη ενέργεια, όπως, π.χ., η συμμετοχή του σε botnet (βλ. και deltaHacker 011, σελ. 12). Η διαφορά του με τον ιό είναι πως, κατά κανόνα, δεν αλλοιώνει άλλα προγράμματα ή αρχεία, ενώ δεν χρειάζεται και κάποιο πρόγραμμα-ξενιστή.

Δούρειος Ίππος (Trojan Horse). Έρχεται μεταμφιεσμένο ως ένα χρήσιμο πρόγραμμα, το οποίο όμως εμπεριέχει κρυφές, κακόβουλες δυνατότητες. Ένας δούρειος ίππος είναι κατά κανόνα μη αυτό-αναπαραγόμενος, αφού σε αντίθεση με έναν ιό ή ένα σκουλίκι ο στόχος του δεν είναι να εξαπλωθεί σε άλλους υπολογιστές.

Spyware. Είναι λογισμικό που εγκαθίσταται στον υπολογιστή χωρίς τη συγκατάθεση του χρήστη. Στέλνει δεδομένα για τις δραστηριότητες του τελευταίου (π.χ., passwords σε διάφορα sites) σε κάποιον κακόβουλο χρήστη, μέσω Internet.

Adware. Πρόγραμμα που συγκεντρώνει και στέλνει σε (ημι)παράνομες διαφημιστικές εταιρείες πληροφορίες σχετικές με τη διαδικτυακή δραστηριότητα του χρήστη (π.χ., ποια sites επισκέπτεται συχνότερα

κατά τη διάρκεια της ημέρας). Ξεκινώντας από αυτά τα στοιχεία δημιουργείται ένα προφίλ, με βάση το οποίο εμφανίζονται στο χρήστη στοχευμένες διαφημίσεις.

Backdoor. Πρόγραμμα που επιτρέπει σε κακόβουλο χρήστη την απομακρυσμένη πρόσβαση ή/και διαχείριση του υπολογιστή όπου βρίσκεται εγκαταστημένο. Ένα backdoor ενδέχεται να είναι αυτόνομο πρόγραμμα που χρησιμοποιείται γι' αυτόν ακριβώς το σκοπό. Εναλλακτικά ίσως να πρόκειται για λειτουργία ενσωματωμένη σε κάποιο άλλο πρόγραμμα, το οποίο παρουσιάζεται ως κάποια δήθεν χρήσιμη εφαρμογή. Παράδειγμα: Στα τέλη του περασμένου Σεπτεμβρίου έγινε γνωστό ότι κάποιος φύτεψε στο κορεάτικο mirror του Sourceforge μια εκδοχή του γνωστού εργαλείου phpMyAdmin, η οποία περιείχε backdoor. Πριν ανακαλυφθεί, πολλοί χρήστες κατέβασαν κι εγκατέστησαν τη συγκεκριμένη εκδοχή του phpMyAdmin.

Rootkit. Είναι λογισμικό που χρησιμοποιεί προχωρημένες τεχνικές απόκρυψης, προκειμένου να εισχωρήσει όσο το δυνατόν βαθύτερα στο λειτουργικό σύστημα και ταυτόχρονα να μην είναι δυνατή η ανακάλυψή του. Συνήθως, τα rootkits αντικαθιστούν κρίσιμα αρχεία του συστήματος με αντίστοιχα παραποιημένα, ενώ διαθέτουν και λειτουργικότητα backdoor. Ένα από τα γνωστότερα rootkits ήταν αυτό που τοποθετούσε κάποτε η γνωστή εταιρεία Sony, στα μουσικά CD της. Η δικαιολογία αφορούσε στην αποτροπή της πειρατείας.

Malware analysis

Η ανάλυση ενός αγνώστου δείγματος κακόβουλου λογισμικού, αυτό που ονομάζουμε malware sample, είναι ουσιαστικά η διαδικασία μελέτης και αποσύνθεσης του συγκεκριμένου προγράμματος στα κύρια συστατικά του. Αυτό γίνεται για να μπορέσουμε να δώσουμε απαντήσεις σε ορισμένα κρίσιμα ερωτήματα που ανακύπτουν. Στα ζητήματα που αφορούν σε ένα malware, συγκαταλέγονται τα ακόλουθα.

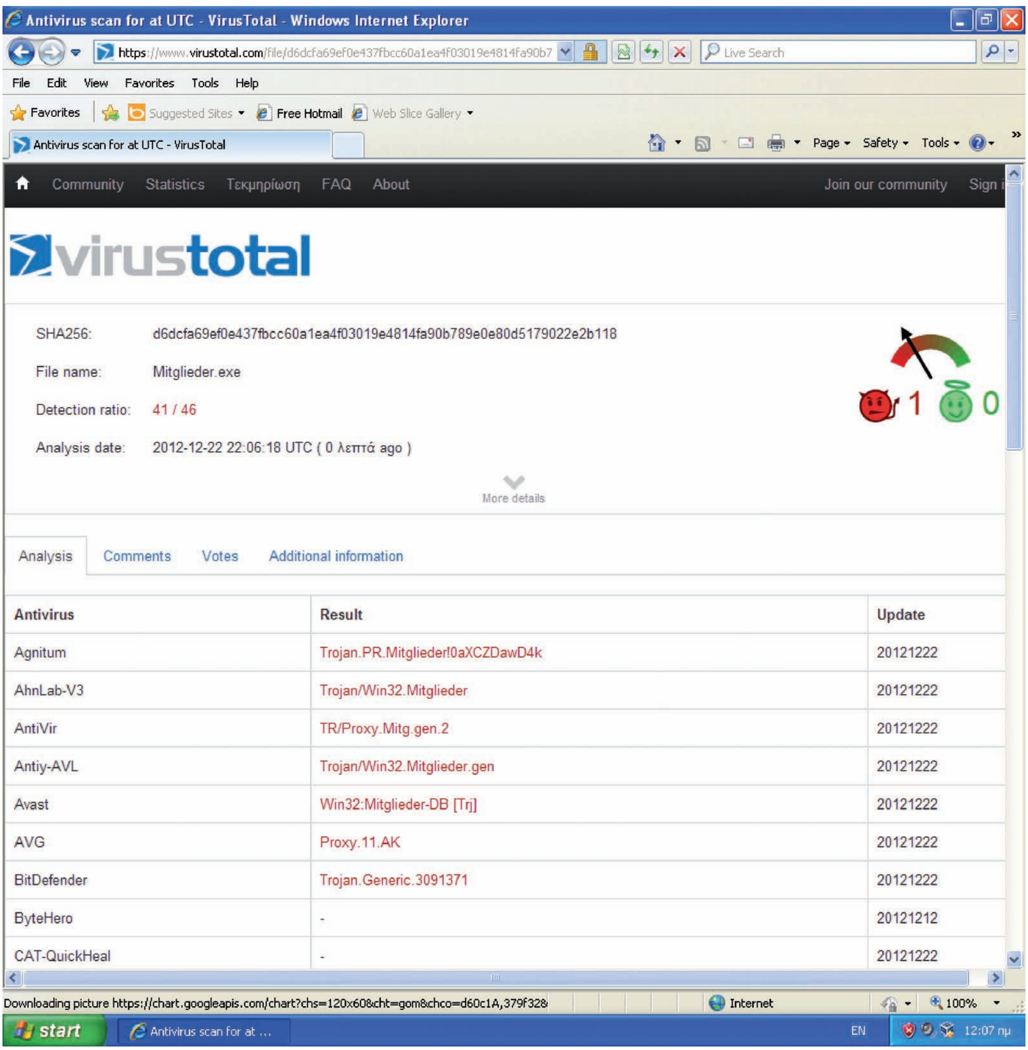
- Ποιος είναι ο σκοπός του malware;
- Πώς εισέρχεται σε ένα σύστημα;
- Τι ενέργειες πραγματοποιεί μετά την εκτέλεση του;
- Πώς μεταφέρεται —αν μεταφέρεται— από σύστημα σε σύστημα;
- Ποια είναι τα σημάδια στο δίκτυο που μαρτυρούν τη δράση του;
- Ποια είναι τα σημάδια στους υπολογιστές-θύματα, που μαρτυρούν τη δράση του;
- Πότε και πώς δημιουργήθηκε το malware; Βασίζεται σε άλλα, γνωστά malware ή βιβλιοθήκες;
- Χρησιμοποιεί τεχνικές κατά της ανάλυσής του; Αν ναι, ποιες;

Ο στόχος του malware analysis είναι ουσιαστικά η συγκέντρωση επαρκών πληροφοριών, ώστε να μπορέσουμε να αντιμετωπίσουμε το συγκεκριμένο κακόβουλο πρόγραμμα. Οι πληροφορίες αυτές είναι ορισμένα μοναδικά χαρακτηριστικά που προσπαθούμε να ανακαλύψουμε για το κακόβολο δείγμα, τα οποία θα μπορέσουν να χρησιμοποιηθούν ως αναγνωριστικά για εκείνο. Αν, π.χ., ένα spyware στέλνει κάθε φορά την ίδια ακολουθία πακέτων στο Διαδίκτυο κι αυτά έχουν συγκεκριμένα χαρακτηριστικά

γνωρίσματα, τότε αυτή η ακολουθία μπορεί να λειτουργήσει ως «υπογραφή» για το υπό εξέταση spyware. Με αυτή την υπογραφή, μια εταιρεία ασφαλείας θα έφτιαχνε έναν κανόνα που θα ενσωμάτωνε σε antivirus ή σε ένα σύστημα ανίχνευσης επιθέσεων (Intrusion Detection System, IDS). Έτσι, από εκεί κι έπειτα το συγκεκριμένο κακόβουλο λογισμικό θα ανιχνευόταν επακριβώς και θα αντιμετωπιζόταν αποτελεσματικά.

Βέβαια, το παραπάνω εγχείρημα δεν είναι και τόσο απλό. Η διαδικασία της ανάλυσης είναι δυνατόν να διαρκεί βδομάδες ολόκληρες, ακόμη κι όταν επιτελείται από έμπειρους αναλυτές. Αυτό γιατί τα τελευταία χρόνια το κακόβουλο λογισμικό εξελίσσεται διαρκώς. Μάλιστα, μεγάλο μέρος του χρηματικού κέρδους που καρπώνονται οι διάφοροι παράνομοι λειτουργοί του malware, επενδύεται στην περαιτέρω εξέλιξή του. Για τις οικονομικές δραστηριότητες που συντηρούν όλη αυτή τη βιομηχανία κακόβουλου λογισμικού, μπορείτε να διαβάσετε και το άρθρο περί bots και botnets που ξεκινά από τη σελίδα 12, του deltaHacker 011.

Στο site του VirusTotal τα κακόβουλα αρχεία υποβάλλονται προς εξέταση σε 40 και πλέον μηχανές antivirus! Εδώ βλέπουμε τα αποτελέσματα από την εξέταση του trojan horse που αναλύουμε...



Μέσα από την παρούσα σειρά άρθρων φιλοδοξούμε να σας δώσουμε μια καλή ιδέα για το πώς πραγματοποιείται η ανάλυση malware στην πράξη. Αν μη τι άλλο, σε λίγο καιρό θα είστε σε θέση να δίνετε ακριβείς απαντήσεις σε ερωτήματα που αφορούν malware samples — κι εννοούμε ερωτήματα σαν αυτά που παραθέσαμε πριν λίγο.

Η διαδικασία της ανάλυσης κακόβουλων προγραμμάτων αποτελείται κυρίως από δύο φάσεις, οι οποίες δεν πραγματοποιούνται κατ' ανάγκη εν σειρά. Πολλές φορές εξάλλου εκτελούνται κι επαναληπτικά, όπου τα νέα δεδομένα που βγαίνουν στο φως βελτιώνουν διαρκώς την εικόνα που έχουμε για το malware.

1. **Ανάλυση συμπεριφοράς.** Εξετάζουμε και καταγράφουμε το πώς αλληλεπιδρά το κακόβουλο δείγμα με το περιβάλλον του (με το σύστημα αρχείων, το registry, το δίκτυο, άλλες ενεργές διεργασίες του συστήματος κ.ο.κ.).

2. **Ανάλυση κώδικα.** Προσπαθούμε να αποσυνθέσουμε το εκτελέσιμο αρχείο και να κατανοήσουμε τη λειτουργικότητά του, εξετάζοντας τον κώδικα Assembly. Υπενθυμίζεται ότι κατά τη μεταγλώττιση ενός προγράμματος παράγεται από τον compiler ένα πρόγραμμα στη συμβολική γλώσσα Assembly. Αυτή είναι μια γλώσσα πολύ χαμηλού επιπέδου, ακριβώς ένα επίπεδο πάνω από την καθαρή, γλώσσα μηχανής που καταλαβαίνει ο υπολογιστής.

Οι παραπάνω δύο φάσεις αντιπροσωπεύουν την ανάλυση malware σε υψηλό, όπως λέμε, επίπεδο. Πιο πρακτικά, οι φάσεις λαμβάνουν χώρα μέσω δύο τεχνικών: της στατικής και της δυναμικής ανάλυσης. Κάθε μία απ' αυτές έχει το βασικό αλλά και το προχωρημένο της στάδιο. Συνολικά, λοιπόν, μπορούμε να μιλάμε για τέσσερεις διαδικασίες. Στα άρθρα μας θα ασχοληθούμε με τα βασικά στάδια των δύο φάσεων. Μάλιστα, σε λίγο θα πραγματοποιήσουμε μια βασική, στατική ανάλυση ενός κακόβουλου δείγματος, παρουσιάζοντας αναλυτικά τη διαδικασία αλλά και τα χρησιμοποιούμενα εργαλεία.

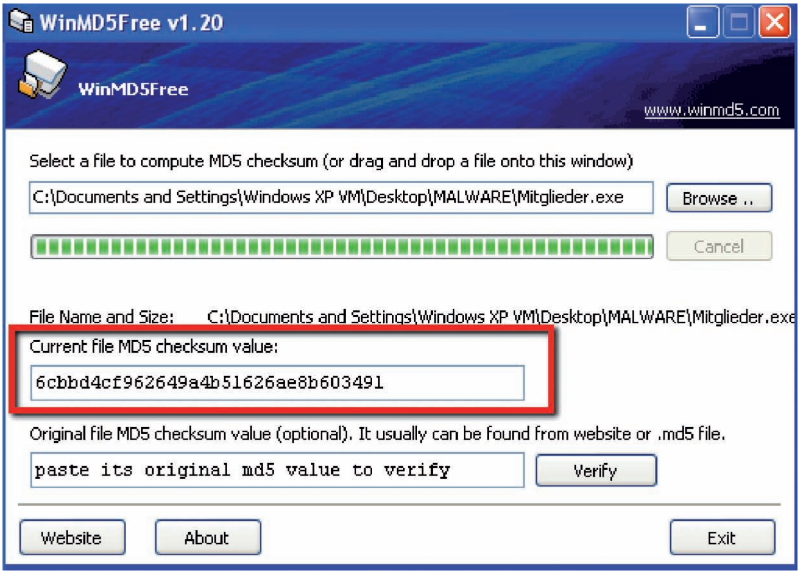
Το εργαστήριο και τα σύνεργά μας

Για τη διενέργεια της διαδικασίας ανάλυσης θα δουλέψουμε σε εικονική μηχανή (Virtual Machine, VM), σαν εκείνες που μπορούμε εύκολα να κατασκευάζουμε με εφαρμογές όπως το VMware Workstation/Fusion ή το VirtualBox. Οι λόγοι για αυτή την επιλογή είναι πολλοί. Οι δύο πλέον προφανείς, είναι οι ακόλουθοι: **α)** θα χρειαστεί σίγουρα να εκτελέσουμε κανονικά το malware sample στον υπολογιστή, οπότε μας βολεύει να το κάνουμε στα πλαίσια μιας απομονωμένης εικονικής μηχανής, χωρίς κίνδυνο για το πραγματικό σύστημα, **β)** είναι πολύ χρήσιμη η δυνατότητα για λήψη στιγμιοτύπων (snapshots), που παρέχουν οι εφαρμογές διαχείρισης εικονικών μηχανών. Ένα snapshot αποθηκεύει, στην ουσία, την τρέχουσα κατάσταση της εκάστοτε μηχανής στον δίσκο του αληθινού υπολογιστή, με όλες τις ρυθμίσεις, τα αρχεία, τις ιδιότητές της κλπ. Έτσι, μπορούμε σε μεταγενέστερο χρόνο να κάνουμε reset την κατάσταση της μηχανής και να επιστρέψουμε σε ένα snapshot της επιλογής μας. Επειδή, όπως αναφέρθηκε, η στατική και η δυναμική ανάλυση πραγματοποιούνται πολλές φορές επαναληπτικά, με αποτέλεσμα να εκτελούμε και το malware άλλες τόσες, μας βολεύει απείρως να μπορούμε εύκολα να ξεκινάμε κάθε φορά από ένα αρχικό, καθαρό στάδιο.

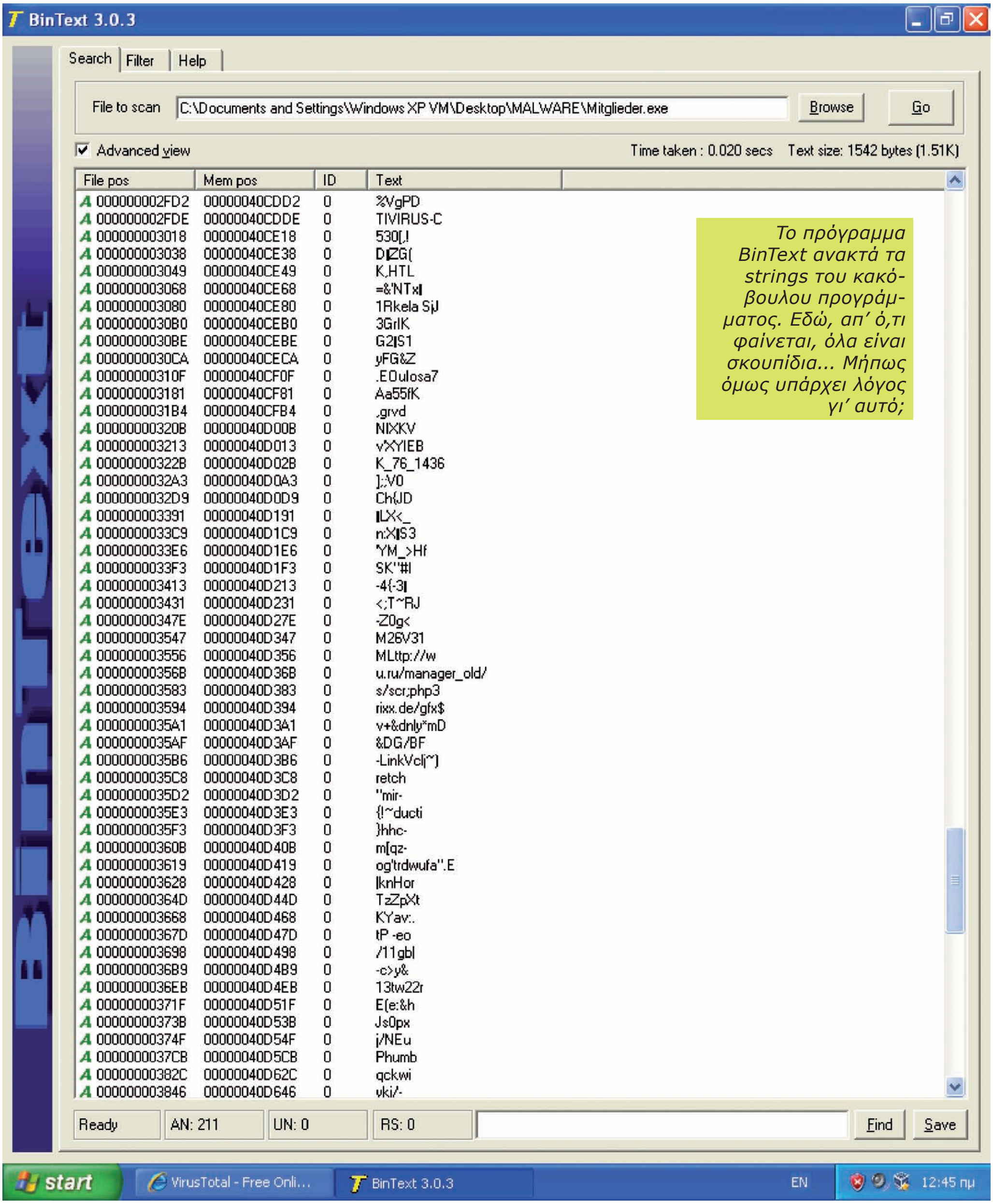
Για τις ανάγκες της δικής μας παρουσίασης δημιουργήσαμε μια εικονική μηχανή με το (δωρεάν) VirtualBox, στην οποία εγκαταστήσαμε τα Windows XP SP3. Αν δεν έχετε εμπειρία από το VirtualBox μπορείτε να διαβάσετε το σχετικό άρθρο που δημοσιεύουμε στο site του περιοδικού και συγκεκριμένα στο http://deltahacker.gr/?p=2667. Πέρα από το λειτουργικό σύστημα που θα εγκαταστήσετε (προτείνεται κάποια έκδοση των Windows XP), θα πρέπει να κατεβάσετε και να ενσωματώσετε στο σύστημα τα ακόλουθα εργαλεία, για τα οποία θα μιλήσουμε εκτενέστερα παρακάτω.

- **WinMD5Free**
  http://winmd5.com
- **Strings v2.5**
  http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx
- **BinText 3.03**
  www.mcafee.com/us/downloads/free-tools/bintext.aspx
- **PEiD 0.95**
  www.softpedia.com/get/Programming/Packers-Crypters-Protectors/PEiD-updated.shtml
- **PE Detective 1.2.1.1**
  www.ntcore.com/pedetective.php
- **Dependency Walker 2.2.6000**
  www.dependencywalker.com

Αφού ενεργοποιήσουμε και τα ίδια τα Windows (αν χρειαστεί) κι αφού έχουμε κατεβάσει και ετοιμάσει όλα τα προαναφερθέντα εργαλεία, μπορούμε να πάρουμε και το πρώτο snapshot της μηχανής. Επιλέγουμε το κεντρικό παράθυρο του VirtualBox και ακολουθούμε τη διαδρομή Snapshots → Take Snapshot. Δίνουμε ένα περιγραφικό όνομα, όπως, π.χ., «Static Malware Analysis Lab».



Το hashing χρησιμοποιείται για την ακριβή ταυτοποίηση κακόβουλων αρχείων. Εδώ βλέπουμε την έξοδο του αλγορίθμου MD5 για το δείγμα που εξετάζουμε, όπως μας τη δίνει το εργαλείο WinMD5Free.



Το πρόγραμμα BinText ανακτά τα strings του κακόβουλου προγράμματος. Εδώ, απ' ό,τι φαίνεται, όλα είναι σκουπίδια... Μήπως όμως υπάρχει λόγος γι' αυτό;

Φέρτε το malware!

Με το εργαστήριό μας έτοιμο, μένει να βρούμε ορισμένα malware samples, για να μας βοηθήσουν στην περιγραφή της διαδικασίας της στατικής ανάλυσης. Επιπρόσθετα θα εκτελέσουμε τα βήματα στην πράξη, ώστε να πάρουμε και απτά αποτελέσματα. Το ερώτημα «πού στο καλό βρίσκω malware» (όταν δεν με βρίσκουν αυτά) έχει δύο απαντήσεις.

Ο ένας τρόπος είναι να πιάσετε τα δικά σας αυτο-εξαπλούμενα κακόβουλα προγράμματα, όπως είναι, π.χ., τα worms ή τα bots. Αυτό μπορεί να γίνει σχετικά εύκολα με το στήσιμο ενός malware honeypot, θέμα που έχει αναπτυχθεί αναλυτικά σε δύο άρθρα του deltaHacker 007. Για παράδειγμα, το Dionaea honeypot που είχε στήσει τότε ο αρθρογράφος, κατάφερε και έπιασε διάφορα ενδιαφέροντα κακόβουλα αρχεία.

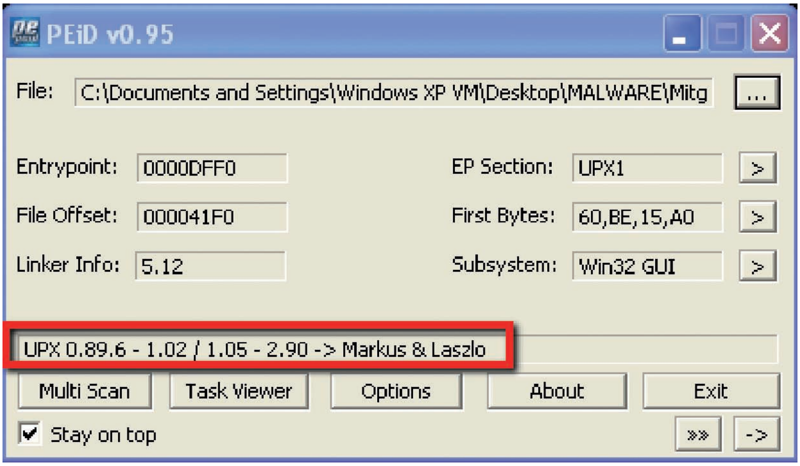
Ένας δεύτερος τρόπος, πιο εύκολος και γρηγορότερος, είναι να βρείτε malware από την κοινότητα των αναλυτών κακόβουλων προγραμμάτων. Υπάρχουν διάφορα αποθετήρια malware, στα οποία σχεδόν καθημερινά ανεβαίνουν νέα δείγματα. Τέτοια αποθετήρια είναι π.χ. το Malware.lu ή το VirusShare.com.

Ένα σχετικό αποθετήριο θα χρησιμοποιήσουμε κι εμείς — συγκεκριμένα το OffensiveComputing.net. Διαθέτει σχεδόν πέντε εκατομμύρια (!) διαφορετικά malware samples και συντηρείται από το Georgia Tech, μέσω της πλατφόρμας Open Malware. Οι μόνοι περιορισμοί για να κατεβάσετε κάποιο δείγμα είναι **α)** να γνωρίζετε κάτι γι' αυτό (όνομα, checksum κ.λπ.) και **β)** να διαθέτετε έναν λογαριασμό Google, για ταυτοποίηση.

Επισκεπτόμαστε λοιπόν τη σελίδα `http://offensivecomputing.net` και για να βρούμε ένα malware sample γράφουμε τη λέξη «trojan», στη μπάρα αναζήτησης πάνω αριστερά. Αμέσως μεταφερόμαστε στην πλατφόρμα Open Malware και βλέπουμε τα αποτελέσματα. Πρόκειται για αρχεία που το όνομα τους εμπεριέχει τη συγκεκριμένη λέξη. Ας δοκιμάσουμε να κατεβάσουμε το πρώτο από αυτά: τον δούρειο ίππο με την ονομασία Mitglieder. Δίνουμε λοιπόν τα στοιχεία του Google account μας και αποσυμπιέζουμε το αρχείο (κωδικός «infected») μέσα στην εικονική μηχανή. Σ' αυτό το σημείο προσποιούμαστε πως δεν ξέρουμε τίποτε για το συγκεκριμένο εκτελέσιμο και συνεχίζουμε όπως θα συνεχίζαμε αν όντως μας έδιδαν ένα άγνωστο αρχείο για ανάλυση.

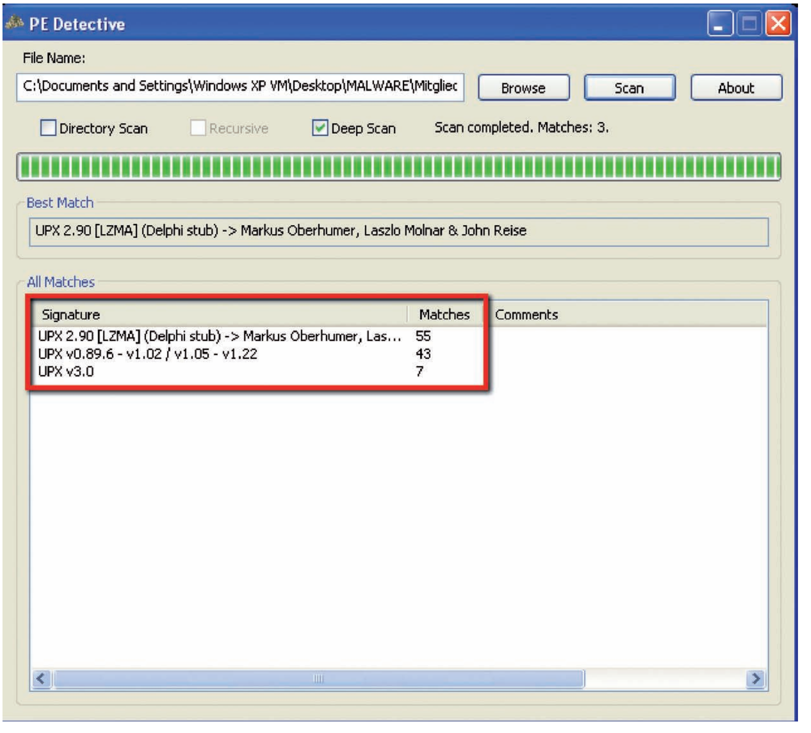
Βασική, στατική ανάλυση malware

Η στατική ανάλυση είναι συνήθως το πρώτο βήμα που εκτελείται κατά την εξέταση ενός κακόβουλου δείγματος. Περιγράφει, στην ουσία, τη διαδικασία ανάλυσης του αρχείου και του κώδικά του, για την εξαγωγή συμπερασμάτων σχετικά με τη λειτουργία του. Το χαρακτηριστικό γνώρισμα της στατικής ανάλυσης είναι ότι κατά τη διάρκειά της *δεν* εκτελεί-



Μπίνγκο! Το PEiD ανίχνευσε ότι το κακόβουλο αρχείο έχει συμπιεστεί με τον γνωστό packer UPX.

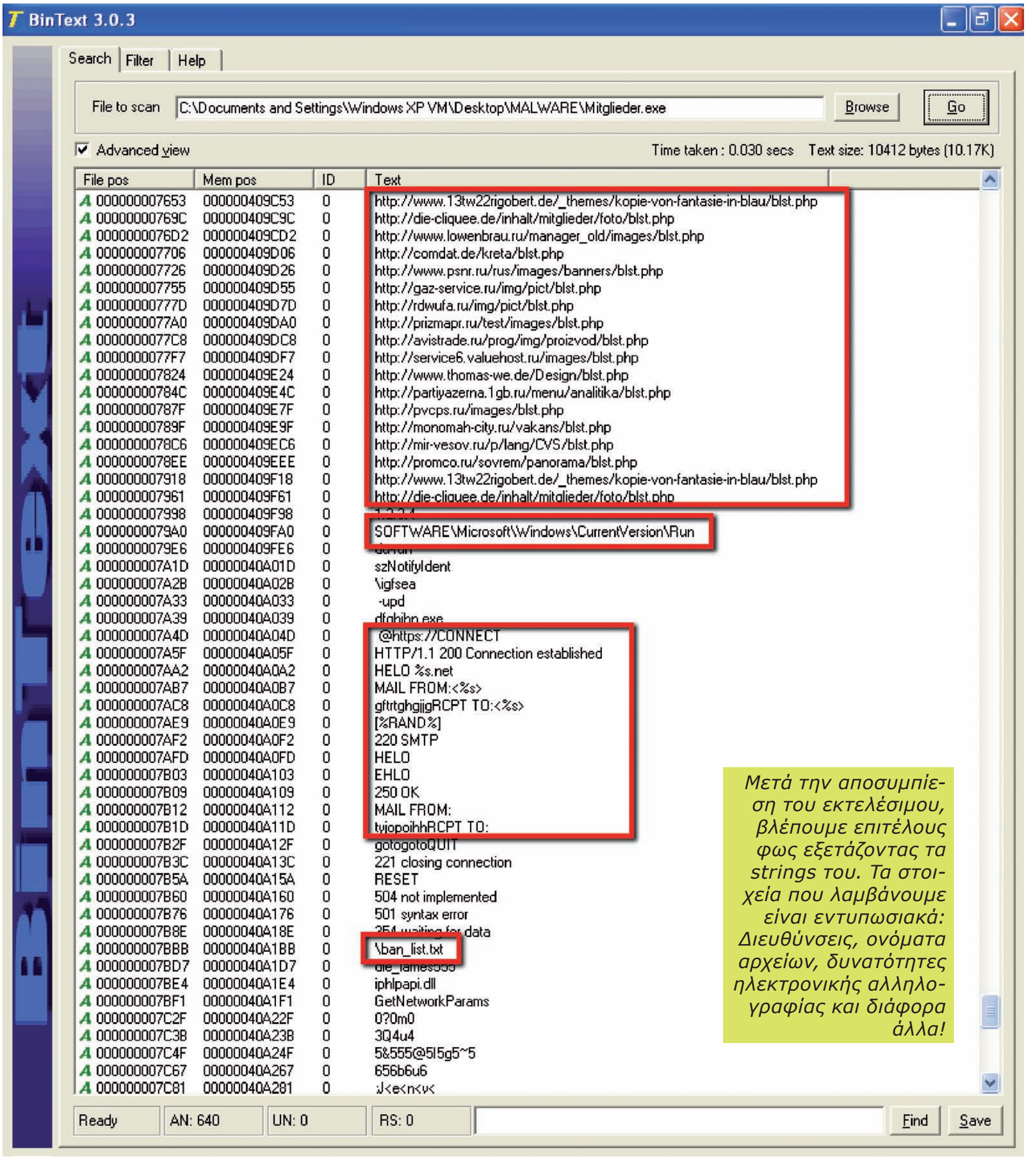
Το PE Detective κάνει κι αυτό καλή δουλειά. Μας επιβεβαιώνει ότι ο προγραμματιστής του trojan horse έχει χρησιμοποιήσει το UPX, ελπίζοντας ότι θα μας κάνει τη ζωή δύσκολη. Αλλά το μόνο που θα κάνει είναι μια τρύπα στο νερό ;)



ται το κακόβουλο αρχείο. Γίνεται δηλαδή προσπάθεια να κατανοηθεί πώς ενεργεί, μόνο με στοιχεία που λαμβάνονται χωρίς αλληλεπίδραση του malware με το σύστημα. Όπως αναφέρθηκε, η στατική ανάλυση έχει δύο στάδια: το απλό και το προχωρημένο. Τα επίθετα αυτά, πέρα των άλλων, παραπέμπουν και στο απαιτούμενο επίπεδο γνώσεων για την ολοκλήρωση καθενός σταδίου. Η προχωρημένη στατική ανάλυση αφορά στην εισαγωγή του εκτελέσιμου αρχείου σε έναν disassembler, όπου παρατηρούμε κι εξετάζουμε των κώδικά του (σε συμβολική γλώσσα Assembly). Στο παρόν άρθρο ασχολούμαστε με το απλό στάδιο, το οποίο είναι πολύ ευκολότερο και μπορεί να πραγματοποιηθεί από τον καθέναν, χωρίς πολλές ειδικές γνώσεις.

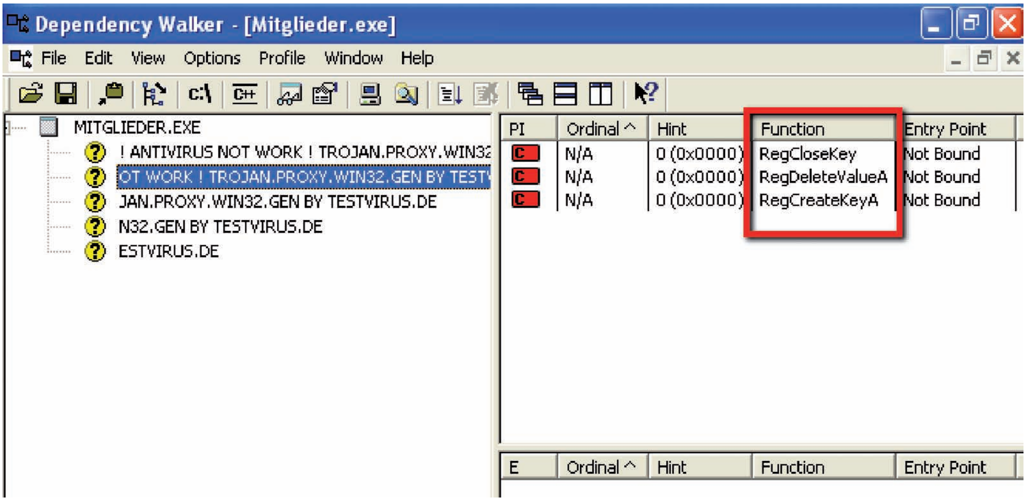
Βήμα 1. Σάρωση του αρχείου από μηχανές antivirus

Όταν εκτελούμε τη στατική ανάλυση, ένα χρήσιμο πρώτο βήμα είναι να περάσουμε το κακόβουλο αρχείο από διάφορες μηχανές antivirus για να δούμε αν έχει πρότερα αναγνωριστεί και καταχωρηθεί από εταιρείες ασφαλείας. Τα προγράμματα antivirus βασίζονται σε δύο τεχνικές. Η πρώτη αναφέρθηκε και προηγουμένως κι έχει να κάνει με τις λεγόμενες υπογραφές των κακόβουλων προγραμμάτων. Αν λοιπόν υπάρχει μια τέτοια υπογραφή για το αρχείο μας, τότε η αναγνώριση θα είναι επιτυχής. Η δεύτερη τεχνική είναι η χρήση ευριστικών, όπως ονομάζονται, μεθόδων. Δηλαδή, το antivirus ακόμη κι αν δεν έχει την υπογραφή του malware, μπορεί να καταλάβει ότι κάτι δεν πάει καλά μ' αυτό το πρόγραμμα παρατηρώντας τη συμπεριφορά του και συγκρίνοντάς τη με ορισμένα γνωστά μοτίβα. Το πρόβλημα εδώ είναι ότι υπάρχουν κακόβουλα προγράμματα που λειτουργούν με τέτοιον τρόπο, ώστε να ξεγελούν τις μηχανές antivirus.



Μετά την αποσυμπίεση του εκτελέσιμου, βλέπουμε επιτέλους φως εξετάζοντας τα strings του. Τα στοιχεία που λαμβάνουμε είναι εντυπωσιακά: Διευθύνσεις, ονόματα αρχείων, δυνατότητες ηλεκτρονικής αλληλογραφίας και διάφορα άλλα!

Ας μην ξεχνάμε επίσης και την πιθανότητα να μην υπάρχει αντίστοιχη υπογραφή, στη βάση δεδομένων του όποιου antivirus. Επειδή μάλιστα είναι σύνηθες η μηχανή ανάλυσης κάθε εταιρείας να χρησιμοποιεί διαφορετικές υπογραφές, ονόματα κι αναγνωριστικά για κάθε κακόβουλο πρόγραμμα, ιδανικά θα θέλαμε να σαρώσουμε το αρχείο μας με πολλές μηχανές. Αυτή τη δυνατότητα ευτυχώς την έχουμε πολύ εύκολα, μέσω κάποιων sites όπως είναι το γνωστό VirusTotal (www.virustotal.com). Εκεί, μπορούμε να αποστείλουμε όποιο αρχείο θέλουμε και αυτό θα σαρωθεί από 46 (!) διαφορετικά antivirus. Επισκεπτόμαστε λοιπόν το site κι ανεβάζουμε το εκτελέσιμό μας (το οποίο ονομάσαμε Mitglieder.exe).



Το κακόβουλο πρόγραμμα χρησιμοποιεί αρκετές συναρτήσεις συστήματος, προκειμένου να εκτελέσει τις παράτυπες ενέργειές του. Αυτές μας τις εμφανίζει το Dependency Walker. Βλέπουμε, π.χ., τρεις συναρτήσεις τροποποίησης του registry. Σίγουρα δεν είναι για καλό :/

Αν σε αυτό το σημείο πάρετε κάποιο μήνυμα για την παλαιότητα του Internet Explorer, μπορείτε να εγκαταστήσετε τη νέα έκδοση από τη διεύθυνση `http://bit.ly/ie8winxp`, προσέχοντας όμως να *μην* επιλέξετε την παράλληλη εγκατάσταση των προτεινόμενων προγραμμάτων (αναφέρονται ως "updates", κατά το setup). Κι αυτό γιατί θα εγκατασταθεί κι ένα εργαλείο anti-malware, το οποίο ενδέχεται να δημιουργήσει προβλήματα. Κατόπιν, επαναλάβετε τη διαδικασία υποβολής του κακόβουλου αρχείου στο VirusTotal. Στην περίπτωση μας βλέπουμε ότι 41 από τις 46 μηχανές antivirus έχουν την υπογραφή του, καθώς και ότι το αναγνώρισαν ως malware!

Βήμα 2. Παραγωγή αναγνωριστικού για το malware sample

Ένα κακόβουλο αρχείο συνήθως δεν παίρνει κάποιο "κανονικό" όνομα. Όπως θα παρατηρήσετε κι από τα αποτελέσματα του VirusTotal, έτσι κι αλλιώς διαφορετικές εταιρείες δίνουν διαφορετικά ονόματα στο ίδιο ακριβώς αρχείο. Πώς επομένως μπορούμε να είμαστε σίγουροι ότι όλοι οι εμπλεκόμενοι καταλαβαίνουν το ίδιο, όταν αναφερόμαστε σε ένα συγκεκριμένο κακόβουλο πρόγραμμα; Τη λύση φέρνει η τεχνική του κατακερματισμού ή αλλιώς hashing.

Με πολύ απλά λόγια, το hashing είναι μία μέθοδος κατά την οποία εισάγουμε δεδομένα σε έναν αλγόριθμο (μαθηματική συνάρτηση) κι αυτός, βάσει των δεδομένων εισόδου, επιστρέφει ένα αλφαριθμητικό συγκεκριμένου πάντοτε μήκους. Ένα τέτοιο αλφαριθμητικό (hash) είναι αστρονομικά απίθανο να παραχθεί από δύο διαφορετικές εισόδους, επομένως στην πράξη μπορεί να χρησιμοποιηθεί ως αναγνωριστικό — ή καλύτερα ως ταυτότητα — για το υπό εξέταση malware.

Αλγόριθμοι κατακερματισμού υπάρχουν διάφοροι. Ο γνωστότερος εξ αυτών είναι ο MD5 (Message-Digest Algorithm). Αφού παραχθεί το μοναδικό αλφαριθμητικό με τη χρήση του αλγορίθμου για το κακόβουλο δείγμα μας, μπορούμε κατόπιν να το χρησιμοποιούμε ως "ταμπέλα" και να το διαμοιράσουμε σε άλλους αναλυτές, ώστε να τους βοηθήσουμε στην αναγνώριση και ταυτοποίηση του υπό εξέταση κακόβουλου προγράμματος. Μπορούμε επίσης ν' αναζητούμε το hash σε sites όπως το VirusTotal, προκειμένου να βρίσκουμε ήδη υποβληθέντα αρχεία με το ίδιο hash.

Για τους παραπάνω λόγους, εγκαταστήσαμε στην εικονική μας μηχανή το εργαλείο WinMD5. Αν το τρέξουμε κι επιλέξουμε το Mitglieder.exe (το trojan που κατεβάσαμε προηγουμένως), θα πάρουμε ως έξοδο το αλφαριθμητικό 6cbbd4cf962649a4b51626ae8b603491. Αυτό είναι ένα —πρακτικά μοναδικό, κατά MD5— όνομα, με το οποίο μπορούμε από εδώ και στο εξής να αναφερόμαστε στο συγκεκριμένο malware sample.

Βήμα 3. Αναζήτηση αλφαριθμητικών

Όλα τα προγράμματα συνήθως εμπεριέχουν διάφορα αλφαριθμητικά ή αλλιώς strings. Ένα string είναι στην ουσία μια ακολουθία χαρακτήρων κι ένα πρόγραμμα διαθέτει τέτοια, π.χ., για την εκτύπωση μηνυμάτων στο χρήστη, τη σύνδεση σε ένα συγκεκριμένο site στο Internet ή σε μια διεύθυνση IP, την εκτέλεση ενεργειών σε αρχεία με συγκεκριμένη διαδρομή στο δίσκο κ.ο.κ. Παρόλο που ένα πρόγραμμα έχει μεταγλωττιστεί κι ο αρχικός, πηγαίος κώδικας στη γλώσσα προγραμματισμού που γράφτηκε, δεν είναι πάντα διαθέσιμος, τα strings παραμένουν εντός του εκτελέσιμου (bianry) και μπορούν ν' ανακτηθούν. Αρκετές φορές μπορούμε να πάρουμε χρήσιμες πληροφορίες και να αποκτήσουμε μια καλή εικόνα για το τι περίπου κάνει ένα κακόβουλο πρόγραμμα, αρκεί να εξετάσουμε με προσοχή αυτά τα αλφαριθμητικά. Δυστυχώς, λόγω της φύσης των αλφαριθμητικών και του πώς αυτά αναπαρίσταται από τον υπολογιστή, ένα πρόγραμμα που εξάγει τα strings ενός άλλου αρχείου θα επιστρέψει και πάρα πολλά "σκουπίδια". Τέτοια μπορεί να είναι διευθύνσεις μνήμης, εντολές προς τη CPU κ.ά, που δεν είναι στην ουσία "πραγματικά" strings. Ευτυχώς, αυτά είναι ευκόλως αναγνωρίσιμα και μπορούμε να τα προσπερνάμε γρήγορα.

Για την εξαγωγή των strings εγκαταστήσαμε δύο προγράμματα στην εικονική μας μηχανή: το Strings (κεφαλαίο "S"), που είναι εργαλείο κονσόλας, καθώς και το BinText, που εκτελείται σε παραθυρικό περιβάλλον. Μπορείτε να χρησιμοποιήσετε όποιο σας βολεύει καλύτερα. Στο παρόν πείραμα θα χρησιμοποιήσουμε το δεύτερο. Διαλέγουμε λοιπόν το κακόβουλο αρχείο και πατάμε στο κουμπάκι Go, προκειμένου να εξαγάγουμε όλα τα strings. Αμέσως εμφανίζεται μια λίστα με όλα τα αλφαριθμητικά του προγράμματος. Κάπου εδώ αρχίζει η διαδικασία της διαλογής. Εξετάζουμε τα strings ένα-ένα απορρίπτοντας τα σκουπίδια και κρατώντας οτιδήποτε άλλο φαίνεται ενδιαφέρον. Στη δική μας περίπτωση διαπιστώνουμε το εξής παράξενο: Όλα μας τα strings φαίνεται να είναι σκουπίδια, πέρα από τρία-τέσσερα που κάτι πάνε να πουν, αλλά δεν το καταφέρνουν :) Τι να συμβαίνει, άραγε; Περνώντας στο τέταρτο βήμα, θα το ανακαλύψουμε.

Βήμα 4: Ανίχνευση τεχνικών packing ή obfuscation

Οι προγραμματιστές κακόβουλου λογισμικού χρησιμοποιούν πολύ συχνά ειδικά εργαλεία για την υλοποίηση τεχνικών packing ή obfuscation, ώστε η ανίχνευση ή/και η εκ των υστέρων ανάλυση να δυσχεραίνει. Τα obfuscated προγράμματα είναι εκείνα που ο δημιουργός τους έχει προσπαθήσει να κρύψει τη λειτουργικότητα τους. Τα packed προγράμματα είναι ειδική περίπτωση των obfuscated. Στην ουσία έχουν μπει σε μια "συσκευασία" ή αλλιώς έχουν πακεταριστεί, όπως μαρτυρά και τ' όνομα τους. Ακριβέστερα, έχουν συμπιεστεί με ειδικά εργαλεία που ονομάζονται packers.

Όταν ένα εκτελέσιμο γίνεται packed, στην πραγματικότητα ενσωματώνεται στην αρχή του ένα δεύτερο πρόγραμμα. Δουλειά του είναι η αποσυμπίεση του κώδικα, όταν το πακεταρισμένο αρχείο εκτελείται. Αποτέ-

λεσμα της διαδικασίας του packing είναι να αποκρύπτονται τα εμφανή στοιχεία του κακόβουλου αρχείου, όπως, π.χ., τα strings ή διάφορες κλήσεις συναρτήσεων. Για να μπορέσουμε να επεξεργαστούμε ένα packed executable, θα πρέπει να το αποσυμπιέσουμε μόνιμα. Αυτό γίνεται σε δύο βήματα: **α)** βρίσκουμε με ποιον ακριβώς packer έχει συμπιεστεί, **β)** εφαρμόζουμε πάνω του την αντίστροφη διαδικασία, χρησιμοποιώντας τον αντίστοιχο unpacker. Πολλές φορές αυτό δεν είναι τόσο εύκολο, καθώς ένας προγραμματιστής μπορεί να έχει χρησιμοποιήσει custom packer! Συνήθως όμως οι περισσότεροι απλά χρησιμοποιούν γνωστούς packers που κυκλοφορούν στο Internet.

Τώρα, προκειμένου να ανακαλύψουμε αν το κακόβουλο δείγμα μας είναι packed, θα καταφύγουμε στα εργαλεία PEiD και PE Detective που εγκαταστήσαμε στην εικονική μηχανή προηγουμένως. Χρησιμοποιούμε δύο, ώστε να συγκρίνουμε τα αποτελέσματα μεταξύ τους. Γενικότερα, είναι καλό να χρησιμοποιούμε περισσότερα του ενός εργαλεία για εργασίες σχετικές με την ανάλυση malware: καθένα ενδέχεται να εφαρμόζει διαφορετική μέθοδο ανίχνευσης του χαρακτηριστικού που ψάχνουμε, οπότε συγκρίνουμε τ' αποτελέσματα για να καταλήξουμε σε πιο σίγουρο συμπέρασμα.

Ας περάσουμε όμως στην εκτέλεση των δύο προγραμμάτων. Επιλέγοντας και στα δύο το κακόβουλο αρχείο μας, αμφότερα συμπεραίνουν ότι έχει συμπιεστεί με τον γνωστότερο, ίσως, packer: το UPX. Για να το αποσυμπιέσουμε ώστε να το αναλύσουμε σωστά, θα χρειαστεί να μεταβούμε στον δικτυακό τόπο όπου φιλοξενείται το UPX (http://upx.sourceforge.net) και να κατεβάσουμε το ίδιο το πρόγραμμα (επιλέγουμε το αρχείο upx308w.zip, για Windows). Εξάγουμε τα περιεχόμενα του ZIP κι ανοίγουμε την κονσόλα των Windows (Δώστε Start → Run, στη γραμμή Open γράψτε cmd και πατήστε το [Enter]). Μεταφερόμαστε στον κατάλογο που βρίσκεται το αρχείο upx.exe, το οποίο μόλις εξήγαμε από το ZIP που κατεβάσαμε. Κατόπιν το εκτελούμε με την παράμετρο –d (για decompress) και δίδοντας τη διαδρομή του αρχείου πάνω στο οποίο θα εφαρμοστεί. Παράδειγμα:

```
C:\Documents and Settings\Windows XP VM>cd Desktop

C:\Documents and Settings\Windows XP VM\Desktop>upx.exe –d
MALWARE\Mitglieder.exe
```

Σε ελάχιστα δευτερόλεπτα, το κακόβουλο εκτελέσιμο θα είναι αποσυμπιεσμένο! Όπως τονίσαμε, οι ενέργειες της ανάλυσης δεν εκτελούνται κατ' ανάγκη εν σειρά. Τώρα, π.χ., είναι μια πολύ καλή στιγμή για να επιστρέψουμε στο προηγούμενο βήμα και να εξαγάγουμε ξανά τα strings του προγράμματος. Κάντε το και θα εκπλαγείτε τόσο από τις διαφορές σε σχέση με πριν, όσο και από τα πολύ ενδιαφέροντα στοιχεία που εμφανίζονται: Συναρτήσεις συστήματος που καλεί το πρόγραμμα, ονόματα γνωστών εργαλείων antivirus και διεργασιών που μάλλον προσπαθεί να αποφύγει ή να τερματίσει, ονόματα αρχείων που διαβάζει ή δημιουργεί (π.χ., ban_list.txt), διαδρομές στο registry που πιθανώς τροποποιεί (π.χ., SOFTWARE\Microsoft\Windows\CurrentVersion\Run — μάλλον προσθέτει τον εαυτό του για να εκκινείται αυτόματα), strings που φανερώνουν τη δυνατότητα αποστολής email (π.χ., MAIL FROM:, SMTP HELO κ.λπ.) καθώς και πολλές hard-coded διευθύνσεις ιστοσελίδων, στις οποίες συνδέεται! Πολλά στοιχεία, τα οποία σίγουρα αξίζουν περαιτέρω εξέτασης κι επεξεργασίας.

Βήμα 5. Ανάκτηση βιβλιοθηκών και συναρτήσεων συστήματος

Το τελευταίο βήμα με το οποίο θα ασχοληθούμε στο παρόν άρθρο είναι η ανάκτηση των βιβλιοθηκών (libraries) που χρησιμοποιεί το κακόβουλο πρόγραμμα. Κατά κανόνα, την ώρα της εκτέλεσής του θα χρησιμοποιεί κάποιες συναρτήσεις από το σύστημα, οι οποίες προσφέρουν ειδικές λειτουργίες όπως, π.χ., δημιουργία αρχείου. Αυτές οι συναρτήσεις βρίσκονται συνήθως σε ειδικά αρχεία βιβλιοθηκών με την κατάληξη .DLL (Dynamic Link Library). Οι βιβλιοθήκες αυτές είναι, όπως προσδίδει και η κατάληξή τους, δυναμικής σύνδεσης. Αυτό πολύ απλά σημαίνει ότι το λειτουργικό σύστημα ψάχνει να βρει τα κατάλληλα DLLs δυναμικά, δηλαδή κατά τη φόρτωση ενός προγράμματος που χρησιμοποιεί συναρτήσεις οι οποίες εμπεριέχονται σε αυτά. Όταν ακολούθως το πρόγραμμα κάνει χρήση της εκάστοτε συνάρτησης, αυτή εκτελείται μέσω της αντίστοιχης βιβλιοθήκης. Για να δούμε ποια DLLs φορτώνονται από το κακόβουλο πρόγραμμά μας, θα χρησιμοποιήσουμε την εφαρμογή Dependency Walker που έχουμε ήδη εγκαταστήσει στην εικονική μηχανή.

Δυστυχώς για το δικό μας παράδειγμα, το αρχείο που έχουμε δεν είναι το ιδανικό: Τα DLLs του συστήματος δεν εμφανίζονται, πιθανώς λόγω πρότερης επεξεργασίας του αρχείου πριν ανέβει ως δείγμα στο αποθετήριο. Μπορείτε όμως να δοκιμάσετε οποιαδήποτε άλλο εκτελέσιμο αρχείο (ακόμα και το ίδιο το πρόγραμμα Dependency Walker), ώστε να δείτε τα DLLs που χρησιμοποιεί. Ακόμη όμως κι έτσι, αν πατήσετε πάνω στα μηνύματα που εμφανίζονται στη λίστα αριστερά, κάτω από το όνομα του κακόβουλου αρχείου, βλέπετε στο παράθυρο πάνω δεξιά τις συναρτήσεις συστήματος που χρησιμοποιεί. Το ποιες απ’ αυτές παρουσιάζουν ενδιαφέρον είναι, όπως και στην περίπτωση των strings, θέμα (μικρής) εμπειρίας. Για παράδειγμα, για την περίπτωση που εξετάζουμε μπορούμε να πούμε ότι ενδιαφέρουσες είναι οι συναρτήσεις χειρισμού αρχείων (WriteFile, CreateFileA, CopyFileA), χειρισμού strings (lstrcpyA, lstlenA), και τροποποίησης του registry (RegCreateKeyA, RegDeleteValueA). Και πάλι, όπως βλέπετε, συνδυάζουμε αυτή τη γνώση με τα προηγούμενα στοιχεία που ανακαλύψαμε —κι αναφερόμαστε στα strings εδώ—, για να σχηματίσουμε μια πιο εμπεριστατωμένη εικόνα γύρω από τη συμπεριφορά του υπό ανάλυση malware. Πλέον, ξέρουμε π.χ. ότι το trojan δημιουργεί ένα νέο κλειδί (key) στο registry και συγκεκριμένα στη διαδρομή που είδαμε προηγουμένως, στο BinText.

Επιλογος, αλλά όχι τόσο γρήγορα!

Αυτή ήταν μια πρώτη γεύση από τη διαδικασία ανάλυσης κακόβουλου λογισμικού. Είδαμε το πρώτο στάδιο της στατικής ανάλυσης και τα εργαλεία που συνήθως χρησιμοποιούνται σ’ αυτό. Κάντε μια ανασκόπηση στα ευρήματα που συγκεντρώσατε από τη διαδικασία και θα δείτε ότι, πράγματι, μάθαμε αρκετά για το malware sample που εξετάσαμε. Κι όχι μόνο αυτό, αλλά το κάναμε χωρίς καν να το τρέξουμε! Φανταστείτε λοιπόν πόσα περισσότερα θα καταλάβουμε για τη λειτουργία του όταν το εκτελέσουμε κιόλας. Μη βιάζεστε, όμως. Θα επιστρέψουμε με νέο άρθρο πάνω στο θέμα, περιγράφοντας τη δεύτερη τεχνική ανάλυσης καθώς και διάφορα έξτρα ζητήματα. Προς το παρόν, από εσάς περιμένουμε τα συμπεράσματά σας από την ανάλυση αρχείων της επιλογής σας! Θα τα λέμε από τα γνωστά κανάλια επικοινωνίας: http://deltahacker.gr/forums, https://facebook.com/deltaHacker και φυσικά https://twitter.com/deltaHacker.

Skill: Intermediate

Tags: Python, wxPython, wxGlade, visual programming, visual design, code generation

Python & wxPython: The wxEditor Project!



Python, wxPython, wxGlade: Ώρα για Visual Programming!

Τι καλά που θα ήταν αν όλα τα προγράμματα αποτελούνταν απλώς από ένα μενού, μερικά εικονίδια σε μια γραμμή εργαλείων και ένα παράθυρο! Σας θυμίζει κάτι αυτή η εικόνα; Μα και βέβαια: Είναι ο text editor που έχουμε φτιάξει μέχρι στιγμής, στα πλαίσια της σειράς μας!

by Sonic2000gr

Βέβαια, ο text editor μας έχει και άλλα πράγματα:

- Ένα text control, το οποίο το ταλαιπωρήσαμε αρκετά στο άρθρο του deltaHacker 015. Μέχρι και σε Rich Text το μετατρέψαμε, προκειμένου να ικανοποιεί τις απαιτήσεις μας.
- Κώδικα που συνδέει όλες τις λειτουργίες μεταξύ τους: Ένα χαρακτηριστικό πρόβλημα των προγραμματιστών Visual Basic είναι ότι τα καταφέρνουν μέχρι και τη σχεδίαση του interface. Μετά, ο κόσμος είναι απλά γεμάτος με κουμπάκια που δεν κάνουν τίποτα, καθώς και από άδεια combo boxes. Το δικό μας πρόγραμμα έχει κώδικα — και ανταποκρίνεται σε αυτά που υπόσχεται!
- Μερικούς κοινούς διαλόγους (dialogs) και message boxes, για λειτουργίες όπως αποθήκευση και άνοιγμα. Ο text editor έχει, βεβαίως, και το απαραίτητο About Box.

Ακούμε όμως τους φίλους μας από τον κόσμο της Visual Basic (εκείνους, με τα διακοσμητικά dialogs) να γελάνε μαζί μας: Βλέπετε, μόνο πολύ απλά προγράμματα —τουλάχιστον όσον αφορά στη διεπαφή— περιορίζονται σε τόσο λίγα στοιχεία γραφικών. Αν δεν το πιστεύετε, ανοίξτε το MS Word ή —Θεός φυλάξοι— το LibreOffice, και επιλέξτε κάτι από το μενού. Αμέσως θα δείτε έναν διάλογο —συνήθως ακαταλαβίστικο κι άσχετο με αυτόν της προηγούμενης έκδοσης— να ξεπροβάλλει, προτρέποντάς σας να συμπληρώσετε text boxes, να μετακινήσετε sliders, να γυρίσετε knobs και γενικά να κάνετε ό,τι χρειάζεται, για να καταστρέψετε το κείμενο που γράφατε με κόπο τόση ώρα. (Ευτυχώς, δηλαδή, που το έχετε στο Mercurial[1])

Ναι, τα περισσότερα προγράμματα για το περιβάλλον γραφικών έχουν αρκετά περισσότερα στοιχεία γραφικών, από ένα text control και μερικούς κοινούς διαλόγους. Και καλά, οι φίλοι μας με τη Visual Basic τα φτιάχνουν αυτά εύκολα: Όλη η φιλοσοφία του περιβάλλοντος προγραμματισμού της γλώσσας περιστρέφεται γύρω από τη σχεδίαση της διεπαφής χρήστη, σε σημείο που ο κώδικας να γίνεται κάτι το δευτερεύον. Εμείς, όμως ,με την καθαρή μας Python, τι κάνουμε;

Σχεδίαση Διεπαφής στο wxPython: Μια καθαρή προσέγγιση

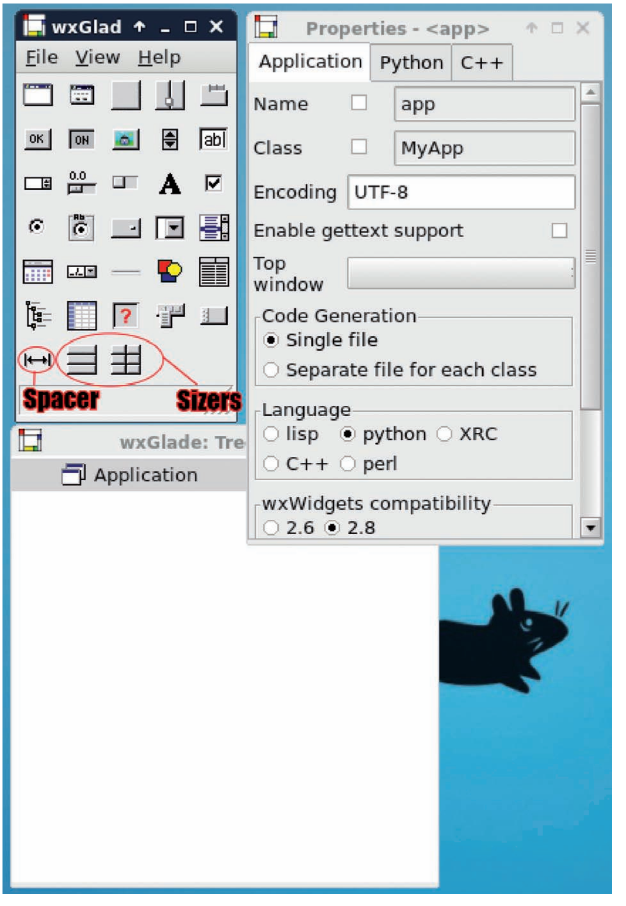
Τι εννοούμε καθαρή προσέγγιση; Για όσους από εσάς έχετε γράψει έστω και λίγο Visual Basic —και μάλλον είστε αρκετοί—, θα έχετε ήδη κατανοήσει πόσα πράγματα μάς κρύβει η γλώσσα. Το είδατε ξεκάθαρα, καθώς γράφαμε τον editor μας. Έπρεπε να γράψουμε binds για να συνδέσουμε events με handler functions. Έπρεπε να έχουμε ειδικά events για να ανανεώνουμε την κατάσταση των εικονιδίων και του μενού μας. Όλο μας το πρόγραμμα είναι ένα μεγάλο event loop, ελαφρά κρυμμένο κάτω από κλάσεις αλλά και πάλι φανερό. Πού είναι αυτά στη VB; Πού βρίσκεται ο κώδικας που δημιουργεί το μενού ή το text control που προσθέσατε στη φόρμα; Κάπου είναι, αλλά κάπου πολύ καλά κρυμμένος! Η VB μάς στερεί τη δυνατότητα να κατανοήσουμε την πραγματική εσωτερική δομή ενός προγράμματος για το περιβάλλον γραφικών. Σημειώστε ότι αυτό δεν συμβαίνει με άλλες γλώσσες της Microsoft. Για παράδειγμα, αν δουλέψετε Visual C++ θα μπορέσετε να δείτε πλήρως τον κώδικα που γράφεται (αυτόματα) για όλα αυτά. Αν το κάνετε, θα σας στείλω και το ζουρλομανδύα δώρο :)

1. Βλέπε deltaHacker 003 και 004.

Στη δική μας, λοιπόν, καθαρή προσέγγιση, θα μπορούμε αμέσως και από την αρχή να σχεδιάζουμε με γραφικό τρόπο τη διεπαφή της εφαρμογής μας σε όποιο επίπεδο θέλουμε (από έναν απλό διάλογο για κάτι που έχουμε ήδη φτιάξει μέχρι μια πλήρη εφαρμογή), καθώς και να βλέπουμε και να επεξεργαζόμαστε τον κώδικα που παράγεται. Αναρωτιέστε γιατί; Μα, επειδή...

Στο δικό μας περιβάλλον ανάπτυξης, ό,τι σχεδιάζουμε μετατρέπεται σε κανονικό κώδικα Python!

Και ποιο είναι το δικό μας περιβάλλον ανάπτυξης; Μα, φυσικά, το wxGlade!



Το περιβάλλον εργασίας του wxGlade, με τα τρία βασικά παράθυρα: εργαλειοθήκη, properties και tree view. Με έντονο, κόκκινο χρώμα, είναι τα αντικείμενα του μαρτυρίου, sizers και spacers! Ξεχάστε ό,τι ξέρατε για συντεταγμένες: στο wxGlade οι θέσεις των controls ρυθμίζονται παίζοντας με αποστάσεις και περιθώρια.

Εγκατάσταση wxGlade

Το wxGlade είναι ένα περιβάλλον για τη σχεδίαση της διεπαφής των εφαρμογών μας. Είναι ένας editor που, αρχικά τουλάχιστον, θυμίζει το περιβάλλον της Visual Basic. Όμως τα φαινόμενα απατούν — και να γιατί:

- Το wxGlade είναι και το ίδιο γραμμένο σε Python και wx.
- Δεν παράγει κρυφό κώδικα, αλλά κανονικό Python code που μπορούμε να χρησιμοποιήσουμε αυτούσιο ή να επεξεργαστούμε/κόψουμε/επικολλήσουμε στο πρότζεκτ μας.
- Δεν πρόκειται για ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Αφορά μόνο στη διεπαφή της εφαρμογής (είναι ένας interface builder, αν θέλετε).

Πρέπει φυσικά να το εγκαταστήσουμε, καθώς δεν έρχεται μαζί με το wxPython. Η εγκατάσταση εξαρτάται από το λειτουργικό μας.

Αναλυτικότερα, σε Debian-based διανομές Linux αρκεί κάτι σαν το παρακάτω:

```
# apt-get install python-wxglade
```

Στο FreeBSD, θα το βάλετε φυσικά από τα ports:

```
# cd /usr/ports/devel/wxGlade
# make install clean
```

Αν πάλι χρησιμοποιείτε το poudriere, θα φτιάξετε το δικό σας πακέτο για το pkgng:

http://deltahacker.gr/?p=7358

Σε Windows, θα επισκεφθείτε τη σελίδα του wxGlade στο Sourceforge (http://sourceforge.net/projects/wxglade) και θα κατεβάσετε έναν απλό installer, από το σύνδεσμο Download. Με την εκτέλεσή του θα έχετε μια γρήγορη και πλήρη εγκατάσταση — ακόμη και με εικονίδιο στο desktop σας.

Αν δουλεύετε σε OS X, από το Download link του Sourceforge θα πάρετε το τελευταίο source tarball. Τη δεδομένη στιγμή είναι το wxGlade-0.6.5, αλλά μέχρι να φτάσει το τεύχος στα χέρια σας ίσως να έχει αλλάξει. Ξέρετε, άλλωστε, πόσο αρέσει στους Open Source developers να πετάνε τη μια έκδοση μετά την άλλη — ειδικά αν έχουν και χρόνο, λόγω των εορτών :)

Αποσυμπιέστε το αρχείο ZIP σε έναν φάκελο της επιλογής σας (π.χ., ~/Desktop/wxGlade-0.6.5). Ανοίξτε το Terminal και δώστε:

```
$ cd ~/Desktop/wxGlade-0.6.5
$ sudo python setup.py install
```

Μπορείτε να εκτελέσετε το wxGlade από το τερματικό σας, γράφοντας, απλώς, wxglade. Το πιθανότερο όμως είναι να πάρετε ένα μήνυμα λάθους, καθώς όπως διαπιστώσαμε ο installer δεν καταφέρνει να εντοπίσει σωστά το path στο οποίο βρίσκεται το wxglade.py *μετά* την εγκατάσταση. Αυτό όμως μπορείτε να το λύσετε ως εξής:

```
$ sudo ln -s /Library/Python/2.7/site-packages/wxglade /usr/lib/python2.7
```

Τώρα, αν όλα αυτά μπορεί να τα κάνει ο τυπικός Mac user, δεν ξέρω. Οι περισσότεροι ακόμα ψάχνουν που είναι το Terminal. Αλλά εσείς το έχετε πρώτο εικονίδιο στο dock, έτσι δεν είναι; (Σ.τ.Ε. You bet :D)

Γνωριμία με το περιβάλλον

Όπως τονίσαμε ήδη, το wxGlade είναι interface builder και όχι πλήρες IDE. Μη περιμένετε λοιπόν να το χρησιμοποιήσετε για να γράψετε πλήρως την εφαρμογή σας. Απλά, θα σχεδιάσετε τη διεπαφή, θα πάρετε τον κώδικα που παράγεται και θα τον βάλετε στην εφαρμογή — την οποία θα συνεχίσετε να δουλεύετε στον αγαπημένο σας text editor.

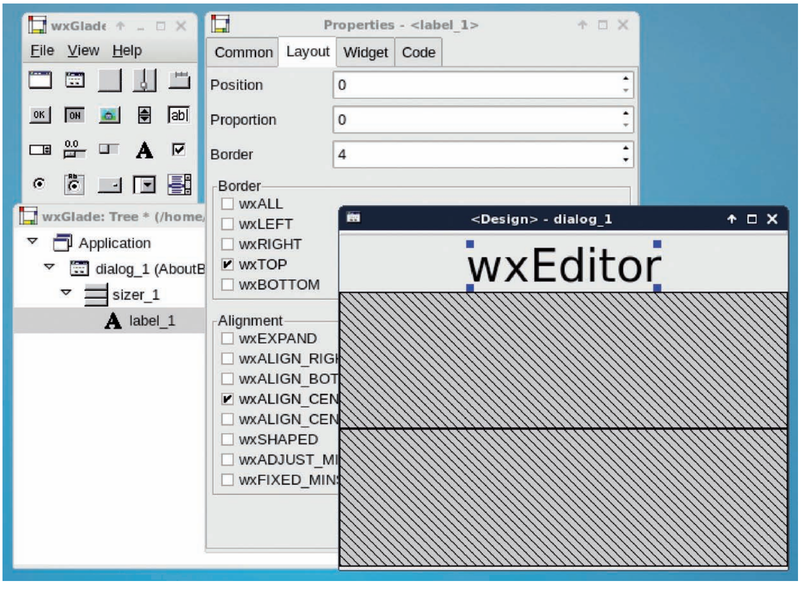
Ξεκινώντας, το wxGlade θα σας δείξει τρία παράθυρα:

- Την εργαλειοθήκη με τα controls που μπορείτε να τοποθετήσετε πάνω στις φόρμες/παράθυρα/frames/διαλόγους ή όπως αλλιώς τα ονομάσουν αύριο, της εφαρμογής σας.
- Το παράθυρο Tree, που δείχνει την ιεραρχία των αντικειμένων.
- Το παράθυρο Properties, το οποίο μας επιτρέπει να ρυθμίσουμε τις ιδιότητες των αντικειμένων που σχεδιάζουμε καθώς και κάποιες άλλες, γενικές για την εφαρμογή μας και τον κώδικα που θα παραχθεί.

Τα δύο πρώτα εργαλεία στην εργαλειοθήκη είναι τα Frame και Dialog. Αν θυμάστε, στον text editor έχουμε ένα πολύ πρόχειρο dialog για το About Box. Μάλλον είναι η ώρα να φτιάξουμε κάτι καλύτερο. Ευκαιρία, λοιπόν, να δοκιμάσουμε και το wxGlade!

Δημιουργία dialog

Πιέστε το δεύτερο εικονίδιο από την εργαλειοθήκη (Dialog). Αμέσως θα εμφανιστεί ένας διάλογος που θα σας ρωτάει για τον ...διάλογο που θα



Για να ρυθμίσουμε την απόσταση των controls μεταξύ τους, εκτός από τα sizers έχουμε και τα borders. Στο παράδειγμά μας έχουμε ενεργοποιήσει μόνο το wxTop, προκειμένου να απομακρύνουμε λίγο το Static Text από το περιθώριο του Frame. Ό,τι δεν μπορούμε να πετύχουμε με sizers και περιθώρια το προσπαθούμε με spacers. Μιλάμε για πολλά ευχάριστα απογεύματα!

φτιάξετε. Επιλέξτε wxDialog. Στο Class σβήστε το (αστείο) MyDialog και γράψτε AboutBoxDialog. Πιέστε OK. Ο κενός διάλογος θα εμφανιστεί στην οθόνη σας.

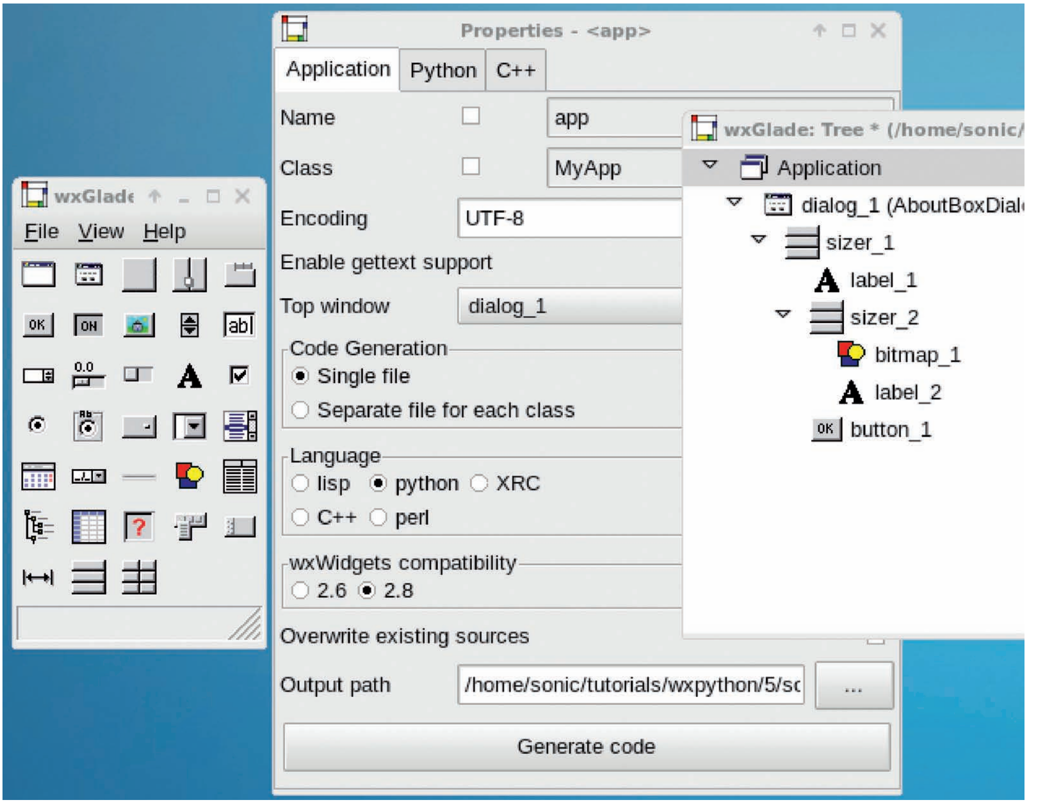
Μέσα στην τρελή χαρά, τρέχετε να πάρετε ένα πλήκτρο (το OK από την εργαλειοθήκη, πρώτο εικονίδιο στη δεύτερη σειρά) και να το βάλετε πάνω στον διάλογο. Αμ, δε! Όσο και να τραβάτε, όσα κλικ και να κάνετε το πλήκτρο δεν θέλει. Μα, γιατί; Στο κάτω–κάτω έτσι το κάνατε πάντα, στη Visual Basic!

Θα σας μαλώσουμε: εδώ δεν είναι Visual Basic, δεν δουλεύουν τα πράγματα με τον τρόπο της. Βλέπετε, στη VB τα controls τα βάζουμε σε όποια θέση θέλουμε πάνω στη φόρμα και υπάρχουν ιδιότητες (X και Y) που δείχνουν στις θέσεις αυτές. Στο wxPython, όμως, τα πράγματα είναι κάπως διαφορετικά.

Δείτε στην τελευταία γραμμή της εργαλειοθήκης: τα δύο τελευταία (και φαρμακερά) εικονίδια, είναι τα λεγόμενα sizers: BoxSizer και GridSizer. Πριν ξεκινήσουμε να βάζουμε οτιδήποτε πάνω στο Frame ή στο Dialog που φτιάχνουμε, πρέπει να βάλουμε οπωσδήποτε ένα από αυτά. Το wxGlade μάς βάζει αυτόματα ένα, όταν φτιάχνουμε Frame (το πρώτο εικονίδιο), αλλά όχι όταν φτιάχνουμε Dialog. Πιέστε λοιπόν το BoxSizer και κάντε κλικ πάνω στο διάλογό μας. Θα ερωτηθείτε για κάποια μυστήρια πράγματα:

- Slots: Γράψτε 3
- Orientation: Επιλέξτε Vertical

Πιέστε OK. Τι είναι αυτό; Το παράθυρο χωρίστηκε σε τρεις περιοχές (slots) κατά ύψος, με οριζόντιες γραμμές. Σωστά αντιλαμβάνεστε πως αν είχατε επιλέξει Horizontal θα χωρίζονταν σε τρεις περιοχές κατά πλάτος, με τρεις κάθετες γραμμές. Ναι, δεν είναι τυπογραφικό λάθος! Θα σκέφτεστε: «κάθετες γραμμές ίσον horizontal, οριζόντιες γραμμές ίσον vertical». Είναι πιο εύκολο έτσι.



Η μαγική στιγμή που κάποιος άλλος γράφει κώδικα για μας: Έτοιμοι να πιέσουμε το Generate Code και το wxGlade θα γράψει ολοκάθαρη, στοιχισμένη Python, έτοιμη να ενσωματωθεί στο πρόγραμμα μας!

Ελπίζουμε να ανακαλύψατε ήδη ότι το παράθυρο properties άλλαξε για να δείξει τις ιδιότητες του sizer (τίποτα σπουδαίο, όμως), καθώς και ότι το παράθυρο Tree μάς επιτρέπει πλέον να διαλέξουμε όποιο αντικείμενο θέλουμε, από την ιεραρχία των αντικειμένων (Application → Dialog → Sizer), για να αλλάξουμε τις ιδιότητές του.

Ας βάλουμε τώρα ένα Label (Static Text), στο πρώτο slot του sizer. Το Static Text είναι το «A», στην εργαλειοθήκη. Μόλις το βάλετε πάνω θα διαπιστώσετε κάτι περίεργο: το slot μίκρυνε όσο χρειάζεται, για να χωρέσει ακριβώς το Static Text που βάλαμε. Μην τρομοκρατείστε — όχι ακόμα, δηλαδή.

Στο παράθυρο Properties επιλέξτε το tab ονόματι Widget και στο Label γράψτε «wxEditor», σβήνοντας το (και πάλι αστείο) label_1, που γράφει. Στο tab με όνομα Common ενεργοποιήστε το Font και κάντε κλικ στις τρεις τελείες, για να βάλετε μεγαλύτερα γράμματα. Εμείς επιλέξαμε μέγεθος 32.

Αν σας τρώει η περιέργεια για το πώς θα φαίνεται μέχρι στιγμής ο διάλογός μας, επιλέξτε από το Tree το dialog_1 και από το Properties, στο Common tab, πιέστε το Preview.

Ωχ, άλλη μια έκπληξη! Ο διάλογός μας είναι ίσα–ίσα το Static Text που γράψαμε. Όλο εκπλήξεις, αυτό το wxGlade. Πίσω στο Common tab του dialog_1, ενεργοποιήστε το size και βάλτε τις τιμές 400, 300 (default).

Ωραία! Το preview τώρα δείχνει πιο φυσιολογικό. Αλλά εμείς θέλου-

με το κείμενο μας να εμφανίζεται κεντραρισμένο οριζόντια, εκεί ψηλά στο πρώτο slot που βρίσκεται. Αν δείτε τις ιδιότητες του Static Text, θα βρείτε το wxALIGN_CENTER_HORIZONTAL στο Layout tab, αλλά και το wxALIGN_CENTRE στο widget tab. Αν εξαιρέσουμε ότι το «κέντρο» πρώτα το γράφουν όπως οι φίλοι μας οι Άγγλοι και μετά όπως οι φίλοι μας οι Αμερικάνοι, τι διαφορά έχουν; Είναι πολύ απλό: Αν το control σας είναι μεγαλύτερο από το μέγεθος του κειμένου που περιέχει (σε μας δεν συμβαίνει), θα κεντράρετε το κείμενο μέσα στο control, από το widget tab. Αν από την άλλη θέλετε να κεντράρετε το ίδιο το control σε σχέση με το slot που το περιέχει, χρειάζεστε την επιλογή από το Layout tab. Αυτό ακριβώς που σας βλέπω να πιέζετε τώρα, που διαβάζετε αυτές τις γραμμές!

Και εντάξει το κεντράρισμα. Τι γίνεται αν θέλουμε να το μετακινήσουμε λίγο πάνω ή κάτω; Στο Layout tab του Static Text, εντοπίστε το πεδίο Border. Αλλάξτε την τιμή του σε 4. Δεν έγινε τίποτα. Για επιλέξτε όμως και το wxTop, από κάτω...

Πιστεύουμε έχετε πιάσει το νόημα: Στο wxGlade δεν υπάρχουν θέσεις X και Y για τα controls: Η γενική θέση ρυθμίζεται με sizers και αυξάνοντας κατάλληλα τα περιθώρια των controls. Είναι λίγο παράξενο αυτό και ενδεχομένως χρειάζεται μια κάποια προετοιμασία και αρκετά previews.

Αν μετανιώσατε που βάλατε το Static Text στο πρώτο slot, μπορείτε να αλλάξετε την τιμή Position: Το 0 είναι το πάνω slot, το 1 το μεσαίο, το 2 το κάτω. Στο slot 1 θα βάλουμε το εικονίδιο της εφαρμογής μας (σε μεγάλο μέγεθος) και φυσικά την άδεια. Πώς όμως θα βάλουμε σε ένα sizer δύο αντικείμενα;

Μάλλον θα μαντέψατε: Θα χρησιμοποιήσουμε άλλο ένα sizer, horizontal αυτή τη φορά, που θα βάλουμε μέσα στο μεσαίο slot. Θα έχει δύο θέσεις: αριστερά το εικονίδιο, δεξιά το κείμενο μας. Στην εργαλειοθήκη, το εικονίδιο με τα τετράγωνα και τον κύκλο είναι για να βάλουμε στατική εικόνα. Επιλέξτε το και κάντε κλικ στο αριστερό slot. Επιλέξτε για εικόνα το editor.ico. Πιστεύουμε ότι μπορείτε να προχωρήσετε μόνοι σας από εδώ και πέρα: Φτιάξτε αν είναι τον διάλογο όπως φαίνεται στη σχετική εικόνα.

Αν καθώς φτιάχνετε τους διαλόγους σας ανακαλύψετε ότι δεν μπορείτε να πετύχετε ακριβώς τη θέση που θέλετε στα controls παίζοντας με τα borders, υπάρχει και το spacer control: είναι απλά ένας ...κενός χώρος. Το βάζετε πάνω σε ένα slot και ρυθμίζετε κατάλληλα τις διαστάσεις του, ώστε να δημιουργήσετε τις αποστάσεις που θέλετε.

Στην τελική έκδοση του διαλόγου, έχουμε μικρύνει τις διαστάσεις του από 400, 300 σε 340, 200, ενώ έχουμε φυσικά προσθέσει ένα OK Button στο κάτω slot του vertical sizer. Εννοείται ότι έχουμε ρυθμίσει το alignment και έχουμε κάνει το σχετικό παιχνίδι με τα borders.

Στο πλήκτρο OK, όμως, κανονικά αντιστοιχεί και κάποιος κώδικας: Τι θα γίνεται όταν το πιέζει ο χρήστης; Δεν πρόκειται να τον γράψουμε μέσα στο wxGlade. Θα ορίσουμε όμως το όνομα της συνάρτησης που θα καλείται: Στο tab ονόματι Event του πλήκτρου, θα βρείτε το EVT_BUTTON και θα γράψετε δίπλα ένα όνομα για τη συνάρτηση σας. Εμείς την ονομάσαμε OnOKClick. Πρωτότυπο, ε;

Τελειώνοντας, αποθηκεύουμε το project (File → Save As) σε μορφή XML, για να το ανοίξουμε και να το επεξεργαστούμε ξανά στο wxGlade. Πρέπει όμως να δημιουργήσουμε και τον κώδικα σε Python, τον οποίο θα ενσωματώσουμε στο πρόγραμμα μας.

Code Generation

Από το tree view, επιλέξτε το Application. Στα Properties → Application μπορείτε να ορίσετε τη γλώσσα (η Python πρέπει να είναι ήδη επιλεγμένη) αλλά και την έκδοση wx (2.8). Στο output path θα γράψετε ή θα επιλέξετε (με το πλήκτρο δίπλα) το όνομα αρχείου όπου θα γραφτεί ο κώδικας.

Πιέζουμε το Generate Code και πάμε να δούμε τον κώδικα που έγραψε για εμάς το wxGlade. Παρακάτω παραλείψαμε μερικά προφανή κομμάτια, για οικονομία χώρου. Το τμήμα που μας ενδιαφέρει περισσότερο είναι, προφανώς, η κλάση AboutBoxDialog. Δημιουργώντας ένα αντικείμενο από αυτή την κλάση, θα έχουμε στην οθόνη μας το πολυπόθητο about box!

```
class AboutBoxDialog(wx.Dialog):
    def __init__(self, *args, **kwds):
        # begin wxGlade: AboutBoxDialog.__init__
        kwds["style"] = wx.DEFAULT_DIALOG_STYLE
        wx.Dialog.__init__(self, *args, **kwds)
        self.label_1 = wx.StaticText(self, -1, "wxEditor")
        self.bitmap_1 = wx.StaticBitmap(self, -1,"editor.ico", wx.BITMAP_TYPE_ANY))
        self.label_2 = wx.StaticText(self, -1, "by Sonic\na deltaHacker Project\
n(C) 2012\nLicensed under the BSD license")
        self.button_1 = wx.Button(self, -1, "OK")
        self.__set_properties()
        self.__do_layout()
        self.Bind(wx.EVT_BUTTON, self.OnOKClick, self.button_1)
        # end wxGlade
    def __set_properties(self):
        # begin wxGlade: AboutBoxDialog.__set_properties
        self.SetTitle("wxEditor")
        self.SetSize((340, 200))
        self.label_1.SetFont(wx.Font(32, wx.DEFAULT, wx.NORMAL, wx.NORMAL, 0, ""))
        # end wxGlade
    def __do_layout(self):
        # begin wxGlade: AboutBoxDialog.__do_layout
        sizer_1 = wx.BoxSizer(wx.VERTICAL)
        sizer_2 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_1.Add(self.label_1, 0, wx.TOP | wx.ALIGN_CENTER_HORIZONTAL, 4)
        sizer_2.Add(self.bitmap_1, 0, 0, 0)
        sizer_2.Add(self.label_2, 0, wx.LEFT | wx.TOP, 10)
        sizer_1.Add(sizer_2, 1, wx.EXPAND, 0)
        sizer_1.Add(self.button_1, 0, wx.BOTTOM | wx.ALIGN_CENTER_HORIZONTAL, 6)
        self.SetSizer(sizer_1)
        self.Layout()
        # end wxGlade
```

Την συνάρτηση OnOKClick θα πρέπει προφανώς να την ολοκληρώσουμε εμείς. Το wxGlade μας έγραψε, ωστόσο, το bind, που θα την ενώνει με τον διάλογο μας καθώς και το stub (http://en.wikipedia.org/wiki/Stubroutine) που φαίνεται παρακάτω:

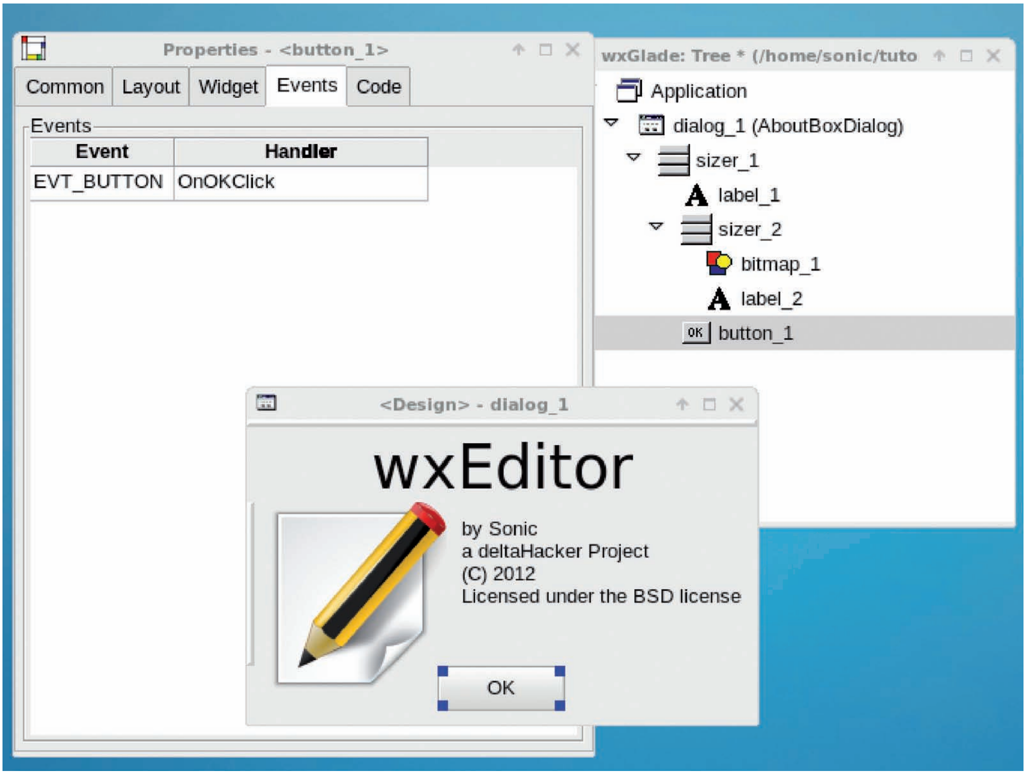
```
def OnOKClick(self, event):
    print "Event handler 'OnOKClick' not implemented"
    event.Skip()
```

Το ωραίο είναι ότι το wxGlade μάς έγραψε και ένα υπόδειγμα συνάρτησης main, ώστε να δοκιμάσουμε απευθείας το πρόγραμμά μας. Εκτελέστε τον κώδικα και θα δείτε το διάλογο στην οθόνη, σαν να προορίζονταν για αυτοτελή εφαρμογή:

```
if __name__ == "__main__":
    app = wx.PySimpleApp(0)
    wx.InitAllImageHandlers()
    dialog_1 = AboutBoxDialog(None, -1, "")
    app.SetTopWindow(dialog_1)
    dialog_1.Show()
    app.MainLoop()
```

Φυσικά, εμείς θα ενσωματώσουμε την κλάση μέσα στο δικό μας stage4.py — και θα την καλέσουμε διαφορετικά.

Ο διάλογος μας, πρακτικά τελειωμένος και έτοιμος να ενσωματωθεί στον text editor. Μια τελευταία πινελιά: Οφείλουμε να ορίσουμε τη συνάρτηση (OnOKClick) που θα καλείται, όποτε ο χρήστης κάνει κλικ πάνω στο OK. Το wxGlade θα γράψει για μας την αντίστοιχη bind και μια άδεια συνάρτηση (stub), που θα πρέπει εμείς να ολοκληρώσουμε.



Ενσωμάτωση στο πρόγραμμα μας

Απλά, προσθέτουμε όλο τον κώδικα της κλάσης AboutBoxDialog στο stage4.py του προηγούμενου άρθρου, και προκύπτει το stage5.py. Φυσικά με μερικές μικρές αλλαγές, τις οποίες μπορείτε να δείτε στο αρχείο που θα κατεβάσετε από το

http://bit.ly/dh016wxgladesamplesrc

Στον υποκατάλογο wxglade θα βρείτε και το αρχείο XML, για να ανοίξετε τον διάλογο στο wxGlade editor.

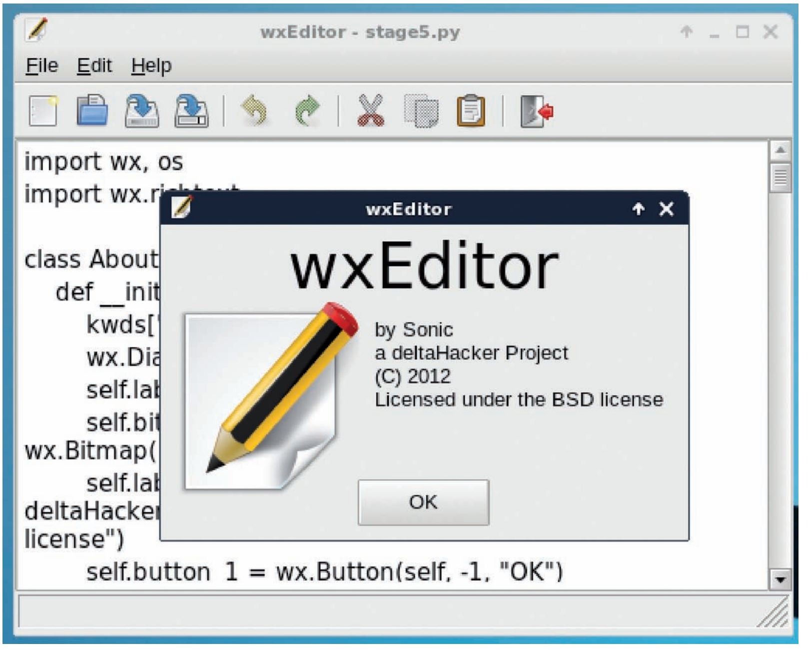
Προφανώς, αφαιρέσαμε τη δοκιμαστική main που μας έγραψε το wxGlade και αλλάξαμε την παλιά μας OnAbout:

```
def OnAbout(self,eventobject):
  aboutdlg = AboutBoxDialog(self, -1, "")
  aboutdlg.ShowModal()
```

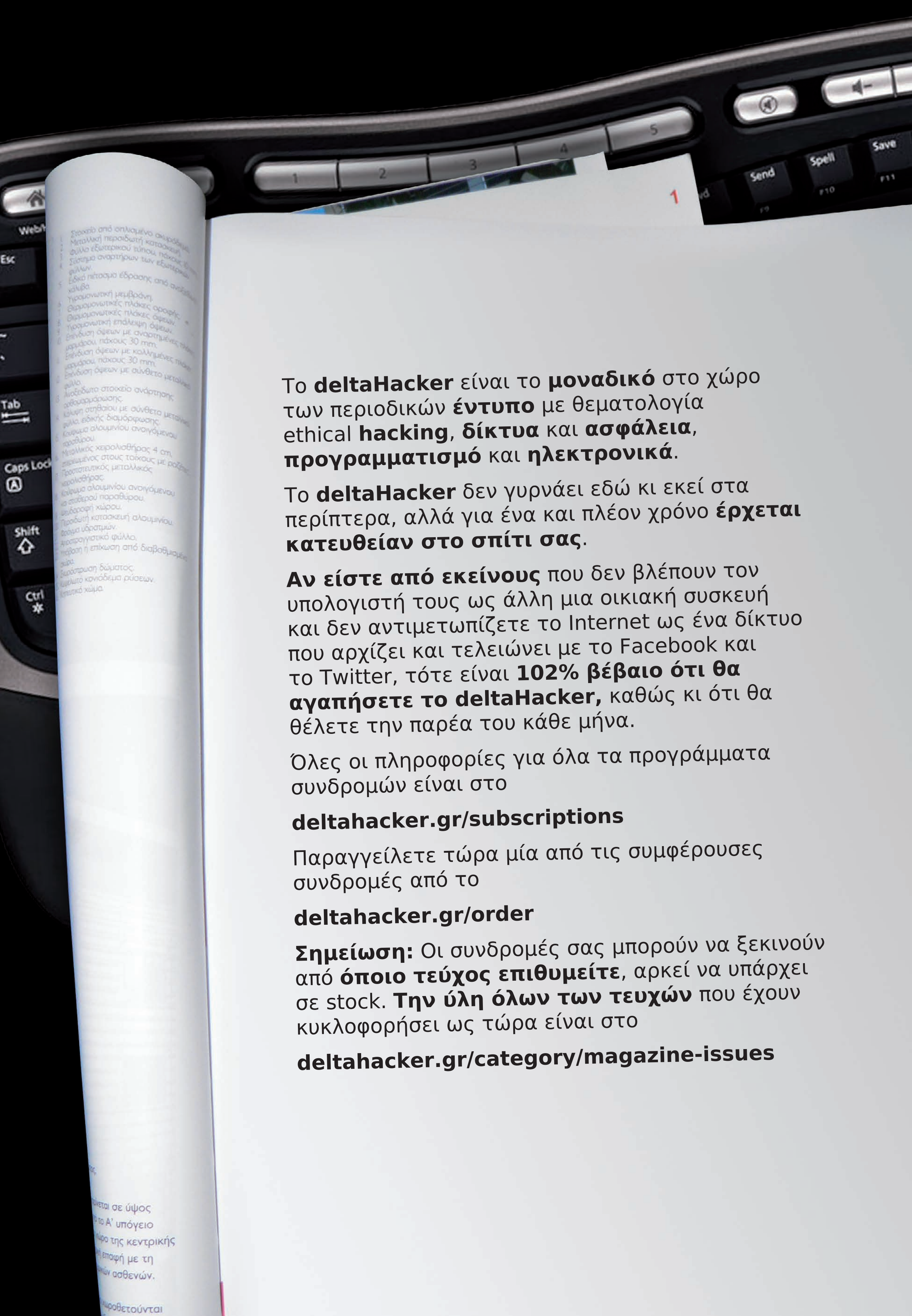
Πρέπει επίσης να γράψουμε κάτι στο OnOKClick. Αλλά ας μην ξεχνάμε ότι το OK απλώς κλείνει τον διάλογο:

```
def OnOKClick(self, event):
  self.Destroy()
```

Τώρα που μάθατε το wxGlade, σας προκαλώ να πειραματιστείτε μόνοι σας! Φτιάξτε πιο πολύπλοκους διαλόγους και ενσωματώστε τους στο πρόγραμμά μας. Τι θα λέγατε για έναν διάλογο με επιλογές; (π.χ., μέγεθος γραμματοσειράς, word wrapping). Θα μπορούσατε ακόμα να μετατρέψετε, σιγά-σιγά, τον editor μας σε έναν επεξεργαστή κειμένου. Το rich text control, εξάλλου, το έχουμε ήδη! Στο επόμενο άρθρο μας θα αφήσουμε τον editor και τα έτοιμα controls και θα ασχοληθούμε με device contexts και απευθείας σχεδίαση. Θα δούμε γραφικά που κανείς Visual Basic programmer δεν έχει φτιάξει, ποτέ!



Ο editor με το έτοιμο brand-spanking-new about Box. Πιο δύσκολο ήταν να βρούμε πώς θα πάνε τα spacers, sizers και borders, παρά να γράψουμε τις πέντε γραμμές κώδικα για να λειτουργήσει!


1
Το deltaHacker είναι το μοναδικό στο χώρο των περιοδικών έντυπο με θεματολογία ethical hacking, δίκτυα και ασφάλεια, προγραμματισμό και ηλεκτρονικά.
Το deltaHacker δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για ένα και πλέον χρόνο έρχεται κατευθείαν στο σπίτι σας.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι 102% βέβαιο ότι θα αγαπήσετε το deltaHacker, καθώς κι ότι θα θέλετε την παρέα του κάθε μήνα.
Όλες οι πληροφορίες για όλα τα προγράμματα συνδρομών είναι στο
deltahacker.gr/subscriptions
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
deltahacker.gr/order
Σημείωση: Οι συνδρομές σας μπορούν να ξεκινούν από όποιο τεύχος επιθυμείτε, αρκεί να υπάρχει σε stock. Την ύλη όλων των τευχών που έχουν κυκλοφορήσει ως τώρα είναι στο
deltahacker.gr/category/magazine-issues

Skill: Intermediate

Tags: Arduino, volume envelope, volume contour, Euler method, fixed point, unions



Καλούπια για την ένταση!

by Spir@l Evolut10n

Το πρώτο μας πιάνο, αυτό που ξεκινήσαμε να φτιάχνουμε στο deltaHacker 015, είναι γεγονός! Ο ήχος του βέβαια είναι ακατέργαστος κι εδώ που τα λέμε δεν θυμίζει πιάνο, αλλά κάτι σαν ψηφιακή σφυρίχτρα. Σ' αυτό το δεύτερο άρθρο της σειράς μας θα ασχοληθούμε με την ένταση του ήχου και θα δούμε πώς οι μεταβολές της επιδρούν στο άκουσμα μίας νότας. Όπως αντιλαμβάνεστε, το πιανάκι μας πρόκειται να δεχτεί την πρώτη του αναβάθμιση ;)

Το μικρό μας πιάνο διαθέτει ένα μόνο κανάλι ήχου. Με απλά λόγια, κάθε στιγμή μπορεί να αναπαράγει μία μόνο νότα. Αυτό θα αλλάξει στο μέλλον και το πιάνο μας θα μπορεί να παίζει αρκετές νότες, ταυτόχρονα. Παράλληλα, θα αναβαθμιστεί και η ποιότητα του ήχου. Το ψηφιακό πιανάκι μας θα ακούγεται σαν αληθινό πιάνο, ενώ η ίδια η κατασκευή θα μπορεί να εξομοιώσει και τον ήχο διαφόρων άλλων οργάνων. Μη φανταστείτε όμως ότι θα περιοριστούμε σε αυτήν την κατασκευή. Το πιανάκι αποτελεί μόνο την αρχή. Πρόκειται για το όχημα, με τη βοήθεια του οποίου φιλοδοξούμε να παρουσιάσουμε όλες τις βασικές έννοιες και τις αντίστοιχες προγραμματιστικές τεχνικές. Σε αυτό το άρθρο θα μελετήσουμε ορισμένες μεταβολές στην ένταση του ήχου, τις οποίες σπανίως παρατηρούμε ή συνειδητοποιούμε. Πρόκειται για μεταβολές οι οποίες συμβάλλουν στη διαφοροποίηση του ήχου από όργανο σε όργανο και, εν πάση περιπτώσει, επιδρούν καθοριστικά στο άκουσμα του ήχου. Κατά τα γνωστά, η μελέτη μας δεν θα περιοριστεί στη θεωρία. Αφού λύσουμε τα όποια προγραμματιστικά προβλήματα ανακύψουν, θα κάνουμε και το πρώτο βήμα προς την κατεύθυνση βελτίωσης του μικρού μας πιάνου.

Ηχητικό περίγραμμα

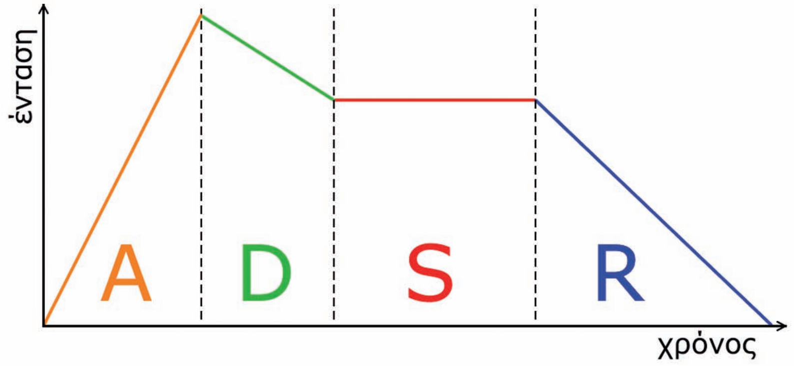
Ας ξεχάσουμε για λίγο το Arduino και τον προγραμματισμό. Ας υποθέσουμε ότι βρισκόμαστε μπροστά σε ένα πιάνο κι ότι κάποια στιγμή πατάμε ένα πλήκτρο, το κρατάμε για λίγο πιεσμένο και στη συνέχεια το αφήνουμε. Δεν μας ενδιαφέρει η νότα που επιλέξαμε κι άρα ούτε η συχνότητα των ηχητικών κυμάτων που δημιουργήθηκαν. Αναλογιστείτε μόνο την ένταση του ήχου που θα ακουστεί. Όπως αντιλαμβάνεστε, ο ήχος δεν θα αρχίσει να ακούγεται *ακαριαία*, ούτε θα πάψει *αμέσως* μόλις αφήσουμε το πλήκτρο. Η ένταση του ήχου θα αρχίσει να αυξάνει και θα κορυφωθεί σε απειροελάχιστο χρόνο, αλλά όχι στιγμιαία. Αντίθετα, όταν αφήσουμε το πλήκτρο η ένταση θα αρχίσει να φθίνει έως ότου εξασθενήσει πλήρως, αλλά με λίγο πιο αργό ρυθμό. Με άλλα λόγια, η ένταση του ήχου θα ακολουθήσει αρχικά μια αύξουσα πορεία και στο τέλος μια φθίνουσα. Όσοι διαθέτουν ευαίσθητο αυτί, είναι σίγουρο ότι θα έχουν παρατηρήσει κάτι ακόμα: Μετά την αρχική κορύφωση της έντασης κι ενώ το πλήκτρο είναι ακόμα πατημένο, η ένταση πραγματοποιεί μια μικρή και σχετικά γρήγορη πτώση, ενώ στη συνέχεια ακολουθεί και πάλι πτωτική πορεία, αλλά με αρκετά μικρό ρυθμό. Η ένταση του ήχου, δηλαδή, μοιάζει να διέρχεται από δύο ακόμα φάσεις: Εκείνη μιας σύντομης και μικρής πτώσης, καθώς κι εκείνη μιας αργής πτώσης. Όλες αυτές οι μεταβολές προσδιορίζουν το λεγόμενο περίγραμμα της έντασης (volume envelope ή volume contour) ενός πιάνου.

Ας υποθέσουμε τώρα ότι φυσάμε (με όλη μας τη δύναμη) σε ένα κόρνο, σε μία τούμπα ή σε κάποιο άλλο χάλκινο πνευστό. Η ένταση του ήχου που θα παραχθεί, ακόμα κι αν καταφέρουμε να κρατήσουμε αμετάβλητη τη ροή του αέρα, σταθερή τη γλώσσα και ακίνητα τα δάκτυλά μας, θα παρουσιάσει αρκετές μεταβολές. Θα αρχίσει να αυξάνει όταν αρχίσουμε να φυσάμε, αλλά σε αντίθεση με το πιάνο, θα αργήσει να κορυφωθεί. Στη συνέχεια θα μειωθεί ελάχιστα και θα διατηρηθεί σχεδόν αμετάβλητη για όσο αντέξουν τα πνευμόνια μας. Μόλις σταματήσουμε το φύσημα η ένταση του ήχου θα αρχίσει να εξασθενεί και θα εξαλειφθεί πλήρως σε *ελάχιστο* χρόνο. Τελικά, φαίνεται ότι ο ήχος του πνευστού χαρακτηρίζεται από το δικό του volume envelope, το οποίο είναι πολύ διαφορετικό από εκείνο του πιάνου.

Αξιοποίηση

Το περίγραμμα της έντασης αποτελεί σημαντικό χαρακτηριστικό του ήχου και συμβάλει στην ηχητική διαφοροποίηση των μουσικών οργάνων. Ακριβώς γι' αυτό, όλα τα synthesizer (σε hardware αλλά και σε software) ενσωματώνουν έναν μηχανισμό που επιτρέπει στο χρήστη να διαμορφώνει το volume envelope κατά βούληση. Με τη βοήθεια του volume envelope generator, όπως ονομάζεται το σχετικό υποσύστημα, ο χρήστης δύναται να προσεγγίσει τον ήχο οποιουδήποτε μουσικού οργάνου ή και να δημιουργήσει εντελώς αλλόκοτους ήχους!

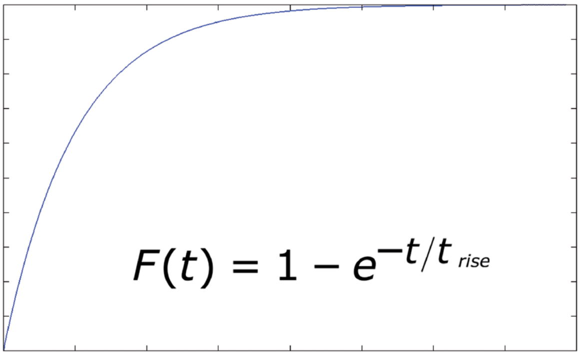
Τα περιγράμματα της έντασης που δώσαμε μόλις, μοιάζουν να χωρίζονται σε τέσσερα στάδια: Σε μια αρχική άνοδο, σε μια σύντομη και μικρή πτώση, σε μια φάση με μικρό ή και μηδενικό ρυθμό πτώσης και τελικά σε μια σχετικά απότομη, τελική κάθοδο. Τα εν λόγω στάδια συμβολίζονται παραδοσιακά με το αρκτικόλεξο ADSR. Πίσω από αυτά τα αρχικά κρύβονται τα μεγέθη Attack, Decay, Sustain και Release, τα οποία χρησιμοποιούνται για την περιγραφή ενός τυπικού volume envelope. Στην πραγματικότητα, το περίγραμμα της έντασης ενός μουσικού οργάνου είναι συνήθως πιο σύνθετο και περιλαμβάνει περισσότερες μεταβολές, ενώ υπάρχουν και περιπτώσεις στις οποίες το περίγραμμα είναι απλούστερο και χαρακτηρίζεται από λιγότερες μεταβολές. Ωστόσο, η απλοποιημένη μορφή με τα τέσσερα στάδια είναι απλή στην υλοποίηση, εύχρηστη και ικανοποιητικότατη για τις ανάγκες ενός synthesizer. Όπως υποψιάζεστε, σε αυτό το άρθρο θα υλοποιήσουμε ένα volume envelope και θα το εφαρμόσουμε στον ήχο του μικρού μας πιάνου. Το δικό μας περίγραμμα δεν θα έχει την τυπική μορφή που περιγράφεται με τα μεγέθη ADSR. Η δική μας υλοποίηση θα είναι ελαφρώς πιο σύνθετη, αλλά θα προσφέρει φυσικότερο ήχο.



Εικόνα 1
Η γενική μορφή του περιγράμματος έντασης (volume envelope), που προσφέρουν τα περισσότερα synthesizer. Τα μεγέθη A (attack), D (decay) και R (release) προσδιορίζουν τη χρονική διάρκεια των αντίστοιχων τμημάτων του περιγράμματος, ενώ το μέγεθος S (sustain) καθορίζει την ένταση του ήχου για το σχετικό αντίστοιχο τμήμα. Αυτή η μορφή του περιγράμματος έντασης, όπως και η μέθοδος περιγραφής με τα μεγέθη A, D, S και R, είναι αρκετά απλοϊκή, αλλά κρίνεται ικανοποιητική για το μέσο synthesizer.

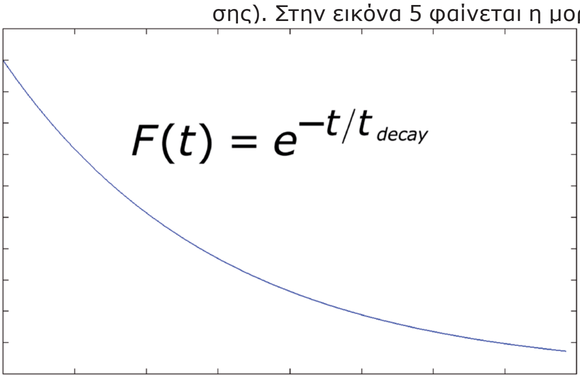
Φυσικό, τεχνηέντως

Για κάθε νότα που παίζουμε στο πιανάκι μας, το πρόγραμμα θα συνεχίσει να παράγει μία μόνο αρμονική ταλάντωση (ένα κύμα σταθερής συχνότητας). Με άλλα λόγια, το ηχόχρωμα του πιάνου μας δεν πρόκειται να βελτιωθεί καθόλου και θα παραμείνει εξαιρετικά φτωχό. Έτσι, για να προσδώσουμε τη μέγιστη δυνατή φυσικότητα στον ήχο του, πρέπει να επιστρατεύσουμε ένα καλά κατασκευασμένο περίγραμμα έντασης. Η πιστή αναπαράσταση του πραγματικού volume envelope ενός πιάνου αποτελεί δύσκολη υπόθεση, ενώ προϋποθέτει το να γνωρίζουμε με ακρίβεια τη μορφή του εν λόγω περιγράμματος. Αυτό που μπορούμε να κάνουμε ωστόσο, είναι να δημιουργήσουμε ένα περίγραμμα έντασης το οποίο θα εξελίσσεται με τρόπο φιλικό προς τις αισθήσεις μας. Αναρωτιέστε τι θα πει



Εικόνα 2
Για τις ανάγκες του δικού μας πιάνου θα κατασκευάσουμε ένα περίγραμμα έντασης με εκθετικές μεταβολές. Η εικονιζόμενη καμπύλη είναι ιδανική για το αρχικό στάδιο του περιγράμματος.

αυτό; Οι αισθήσεις μας είναι προσαρμοσμένες και πιο δεκτικές στις εκθετικές μεταβολές των φυσικών μεγεθών. Έτσι, ένας ήχος με εκθετικά μεταβαλλόμενη ένταση μοιάζει πιο φυσικός για τα αυτιά μας, σε αντίθεση με κάποιον του οποίου η ένταση μεταβάλλεται γραμμικά[1]. Κατά συνέπεια, για να δημιουργήσουμε ένα volume envelope που θα μοιάζει να προέκυψε από κάποια φυσική διαδικασία, αρκεί να αντικαταστήσουμε τα ευθύγραμμα τμήματα με εκθετικές καμπύλες. Για την ταχύτατη αύξηση της έντασης που παρατηρείται στο ξεκίνημα μιας νότας του πιάνου, μπορούμε να επιστρατεύσουμε την καμπύλη που φαίνεται στην εικόνα 2. Αντίστοιχα, για την πτώση της έντασης που παρατηρείται με την πάροδο του χρόνου, μπορούμε να χρησιμοποιήσουμε μια καμπύλη σαν αυτή της εικόνας 3. Όσο για την απότομη εξασθένηση που παρατηρείται όταν αφήνουμε το πλήκτρο, μπορούμε να χρησιμοποιήσουμε πάλι μια παρόμοια καμπύλη, αλλά με μεγαλύτερη κλίση. Τελικά, για να κάνουμε πιο πειστικό το μικρό μας πιάνο, θα κατασκευάσουμε ένα volume envelope σαν αυτό της εικόνας 4. Για τον υπολογισμό των σημείων αυτής της καμπύλης αρκεί να πολλαπλασιάσουμε τις δύο συναρτήσεις, που περιγράφουν τις δύο επιμέρους καμπύλες (εκείνη της εκθετικής αύξησης κι εκείνη της εκθετικής μείωσης). Στην εικόνα 5 φαίνεται η μορφή που θα αποκτήσουν τελικά τα ηχητικά κύματα του πιάνου μας. Όπως βλέπετε, το πλάτος των ημιτονοειδών ταλαντώσεων προκύπτει από την εκθετική συνάρτηση, η οποία περιγράφει το volume envelope.



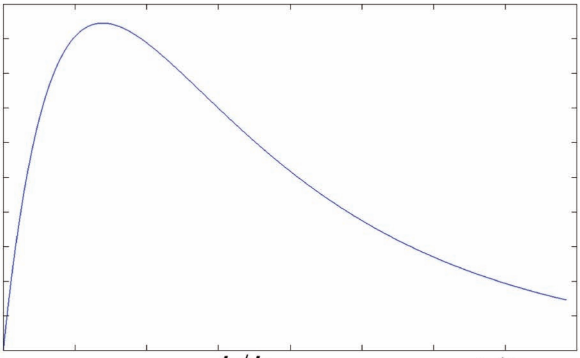
Εικόνα 3
Αυτή η καμπύλη είναι ιδανική για το τελικό στάδιο του περιγράμματος έντασης που κατασκευάζουμε. Σημειώστε ότι και στις δύο περιπτώσεις μπορούμε να μεταβάλλουμε την κλίση της καμπύλης, τροποποιώντας κατάλληλα τις σταθερές t_{rise} και t_{decay}

Ζήτημα χρόνου

Η παραπάνω ιδέα ακούγεται αρκετά καλή και, όπως θα διαπιστώσουμε αργότερα, είναι! Ωστόσο μας φέρνει αντιμέτωπους με ένα πρόβλημα: Πώς θα υλοποιήσουμε τις εκθετικές συναρτήσεις στο πρόγραμμά μας; Οι σχετικές πράξεις πρέπει να πραγματοποιούνται στον ελάχιστο χρόνο, ο οποίος μεσολαβεί ανάμεσα στον υπολογισμό δύο διαδοχικών δειγμάτων του ηχητικού κύματος. Αυτός ο χρόνος είναι εξαιρετικά περιορισμένος κι ως εκ τούτου είναι αδύνατο να χρησιμοποιήσουμε τις έτοιμες λύσεις (τις εκθετικές συναρτήσεις από τη βιβλιοθήκη math, της γλώσσας C). Μόνο κάποιο τέχνασμα κι ενδεχομένως μερικές υποχωρήσεις στην ακρίβεια, θα μας βγάλουν από τη δύσκολη θέση. Ας ξαναρίξουμε μια ματιά στο πρόβλημα...

Οι μεταβολές που θέλουμε να περιλαμβάνει το περίγραμμα της έντασης περιγράφονται από εκθετικές συναρτήσεις. Εμείς όμως δεν ενδιαφερόμαστε για *κάθε* καμπύλη αυτής της μορφής. Ψάχνουμε έναν τρόπο

1. Για παρόμοιους λόγους, στους παλιούς αναλογικούς ενισχυτές χρησιμοποιούνταν λογαριθμικά ποτενσιόμετρα κι όχι γραμμικά!

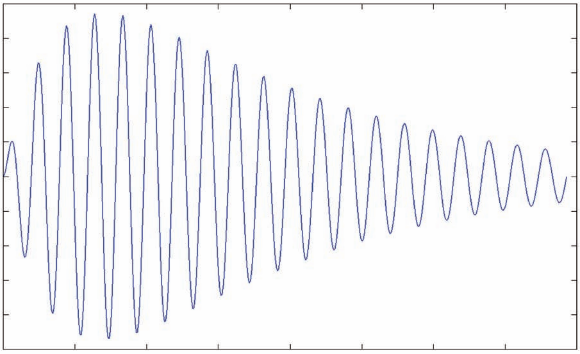


$$F(t) = e^{-t/t_{decay}} * (1-e^{-t/t_{rise}})$$

Εικόνα 4
Το περίγραμμα έντασης που θα χρησιμοποιήσουμε για το πιανάκι μας προκύπτει από το συνδυασμό δύο εκθετικών καμπυλών: Μιας αύξουσας και μιας φθίνουσας. Συγκεκριμένα, η εικονιζόμενη καμπύλη προκύπτει από το γινόμενο των αντίστοιχων εκθετικών συναρτήσεων.

για να υπολογίζουμε τα διαδοχικά σημεία των *συγκεκριμένων* εκθετικών καμπυλών. Αναρωτιέστε τι το συγκεκριμένο έχουν οι δικές μας καμπύλες; Είναι πολύ απλό αλλά εξαιρετικά σημαντικό: Γνωρίζουμε το αρχικό σημείο της καθεμιάς. Με άλλα λόγια, γνωρίζουμε την πρώτη τιμή των αντίστοιχων συναρτήσεων. Κάπου εδώ, οι Μαθηματικοί της παρέας ενδέχεται να διαμαρτυρηθούν: Τι εννοούμε όταν λέμε την «πρώτη τιμή»; Στην περίπτωση της εκθετικής αύξησης, αναφερόμαστε στη στιγμή κατά την οποία ξεκινά ο ήχος. Εκείνη τη στιγμή η έντασή του είναι μηδέν και η τιμή της αντίστοιχης συνάρτησης είναι επίσης μηδέν. Στην περίπτωση της εκθετικής μείωσης, θεωρούμε ότι το φαινόμενο ξεκινά όταν η ένταση έχει φτάσει στη μέγιστη τιμή. Επομένως, η πρώτη τιμή για την αντίστοιχη συνάρτηση ισούται με τη μέγιστη ένταση. Αυτήν την τιμή για την ένταση θα την αποφασίσουμε αργότερα, όταν θα καταπιαστούμε με την υλοποίηση του προγράμματος. Σημασία όμως έχει ότι το εν λόγω μέγεθος είναι γνωστό. Δώστε τώρα βάση: Το πρόγραμμά μας πρέπει κάθε φορά να υπολογίζει την επόμενη τιμή για κάθε μια από αυτές τις συναρτήσεις. Κάθε φορά, δηλαδή, θα γνωρίζουμε την *τρέχουσα* τιμή των συναρτήσεων και θα πρέπει να υπολογίσουμε την επόμενη. Αυτό θα μπορούσε να γίνει πολύ εύκολα, αν γνωρίζαμε το ρυθμό με τον οποίο μεταβάλλεται καθεμία συνάρτηση. Αλλά για μια στιγμή! Μίλησε κανείς για ρυθμό μεταβολής; Ο ρυθμός μεταβολής, ξέρετε, είναι η λεγόμενη παράγωγος...

Μαθηματικοί μπελάδες



Εικόνα 5
Κάπως έτσι θα μεταβάλλεται το πλάτος των ηχητικών κυμάτων του πιάνου μας, όταν εφαρμόσουμε το περίγραμμα έντασης που είδαμε νωρίτερα.

Από δω το πήγαμε, από κει το φέραμε, και τι πετύχαμε; Μπλέξαμε πάλι με τα μαθηματικά! Ας θυμηθούμε πώς φτάσαμε εδώ: Το ζητούμενο είναι να κατασκευάσουμε ένα περίγραμμα έντασης που θα σχηματίζεται από εκθετικές μεταβολές. Για αυτές τις μεταβολές γνωρίζουμε τη μορφή των αντίστοιχων συναρτήσεων και, επιπρόσθετα, γνωρίζουμε τις αρχικές τιμές αυτών των συναρτήσεων. Αν γνωρίζαμε και το ρυθμό με τον οποίο μεταβάλλεται καθεμία, θα ήταν ιδιαίτερα εύκολο να υπολογίσουμε και τη *δεύτερη* τιμή. Στη συνέχεια, γνωρίζοντας αυτήν την τιμή θα μπορούσαμε να υπολογίσουμε την *τρίτη* τιμή και πάει λέγοντας. Αυτό που πρέπει να διερευνήσουμε, λοιπόν, είναι ο τρόπος με τον οποίο μεταβάλλονται οι εκθετικές συναρτήσεις μας. Για το σκοπό αυτό, θα μελετήσουμε την ακόλουθη συνάρτηση:

$$F(t) = e^{-t/t_{decay}}$$

Αυτή η συνάρτηση περιγράφει την εκθετική μείωση. Το μέγεθος t_{decay} αποτελεί μια σταθερά, η οποία ρυθμίζει την ταχύτητα αυτής της μείωσης. Η συγκεκριμένη σταθερά ονομάζεται και σταθερά χρόνου. Όσο πιο μεγάλη τιμή έχει η σταθερά χρόνου, τόσο πιο αργά εξελίσσεται η εκθετική μείωση και αντίστροφα. Ας αφήσουμε αυτές τις παρατηρήσεις όμως κι ας επιστρέψουμε στο θέμα μας. Για να βρούμε το ρυθμό μεταβολής της συνάρτησης, πρέπει να υπολογίσουμε την παράγωγό της:

$$\frac{dF(t)}{dt} = -T_c * F(t)$$

Μάλιστα. Να μια ωραιότατη διαφορική εξίσωση :) Το μέγεθος T_c σχετίζεται με τη σταθερά χρόνου (t_{decay}) και προκύπτει από τη σχέση $T_c = 1 / t_{decay}$. Οι λεπτομέρειες για αυτή τη σταθερά όμως είναι μάλλον αδιάφορες. Το αρχικό μας πρόβλημα, αυτό της υλοποίησης των εκθετικών συναρτήσεων, έχει αντικατασταθεί πλέον από ένα άλλο: Για να υπολογίζουμε κάθε φορά τις διαδοχικές τιμές των εκθετικών μεταβολών, πρέπει να λύσουμε προγραμματιστικά τη συγκεκριμένη διαφορική εξίσωση. Αυτό το πρόβλημα μοιάζει πολύ πιο δύσκολο από το αρχικό, αλλά χάρη στον μεγάλο Μαθηματικό Euler είναι ευκολότερο! Χωρίς να μπούμε σε κουραστικές λεπτομέρειες, θα σας πούμε μόνο ότι θα αξιοποιήσουμε τη λεγόμενη μέθοδο του Euler (Euler method). Οι προϋποθέσεις για την εφαρμογή αυτής της μεθόδου ανήκουν στις «κουραστικές λεπτομέρειες» —και μόλις έκανα air-quotes—, άρα δεν θα μας απασχολήσουν. Εφαρμόζοντας τη λύση του Euler, η παραπάνω εξίσωση μπορεί να γραφεί ως εξής:

$$\frac{F_{n+1} - F_n}{T_s} = -T_c * F_n$$

Το F_n αποτελεί την τρέχουσα τιμή της εκθετικής συνάρτησης, ενώ το F_{n+1} αποτελεί την επόμενη τιμή. Το μέγεθος T_s αποτελεί μια σταθερά, η οποία αντιστοιχεί στη χρονική απόσταση μεταξύ της τρέχουσας και της επόμενης τιμής. Υποσχεθήκαμε όμως ότι δεν θα σας ζαλίσουμε με πολλές λεπτομέρειες. Σημειώστε μόνο ότι και το T_s αποτελεί μια σταθερά. Όπως έχουμε πει, το πρόγραμμά μας θα γνωρίζει πάντα την τρέχουσα τιμή και θα αναζητά την αμέσως επόμενη. Επομένως, θα πρέπει να λύσουμε την εξίσωση ως προς F_{n+1}:

$$F_{n+1} = F_n - T_c * T_s * F_n$$

Δεν ξέρουμε αν το συνειδητοποιήσατε, αλλά το αρχικό μας πρόβλημα έχει ήδη απλοποιηθεί σε τεράστιο βαθμό. Με τη βοήθεια της τελευταίας εξίσωσης, οι περίπλοκες πράξεις για τον υπολογισμό μιας εκθετικής συνάρτησης αντικαθίστανται από μια αφαίρεση και δύο πολλαπλασιασμούς. Το πρόβλημά μας όμως μπορεί να απλοποιηθεί ακόμα περισσότερο!

Και λίγη αλχημεία

Το γινόμενο των T_c και T_s αποτελεί μια σταθερά, η οποία σχετίζεται με τη σταθερά χρόνου της αρχικής εκθετικής συνάρτησης. Επομένως, μπορούμε να αντικαταστήσουμε το γινόμενο με μια τιμή της επιλογής μας, σύμφωνα με την κλίση που θέλουμε να έχει η αντίστοιχη εκθετική καμπύλη. Εδώ πρέπει να σκεφτούμε πονηρά. Το μέγεθος που θα επιλέξουμε θα πολλαπλασιάζεται κάθε φορά με την τρέχουσα τιμής της εκθετικής συνάρτησης. Οι πολλαπλασιασμοί, όμως, σε γενικές γραμμές αποτελούν χρονοβόρες πράξεις. Ξέρετε πότε πραγματοποιούνται γρήγορα; Όταν ο ένας παράγοντας αποτελεί δύναμη του δύο! Βλέπετε, για να διπλασιάσουμε έναν αριθμό αρκεί να ολισθήσουμε όλα τα bits που τον απαρτίζουν κατά μία θέση προς τα αριστερά. Για να τον τετραπλασιάσουμε (2^2) πρέπει να κάνουμε πάλι μια ολίσθηση προς τα αριστερά, αλλά κατά δύο θέσεις. Αντίστοιχα, για να πολλαπλασιάσουμε με το 8 (2^3) πρέπει να κάνουμε μια ολίσθηση προς τα αριστερά κατά τρείς θέσεις και πάει λέγοντας. Όταν διαιρούμε με δυνάμεις του δύο η κατάσταση δεν αλλάζει δραματικά. Απλά, η κατεύθυνση των ολισθήσεων αντιστρέφεται και πραγματοποιείται προς τα δεξιά. Έχοντας αυτά κατά νου, μπορούμε να αποδώσουμε στο γινόμενο T_c * T_s μια τιμή της ακόλουθης μορφής:

$$T_c * T_s = 2^{-k} = \frac{1}{2^k}$$

Συνδυάζοντας τις τελευταίες δύο εξισώσεις καταλήγουμε στην ακόλουθη:

$$F_{n+1} = F_n - 2^{-k} * F_n$$

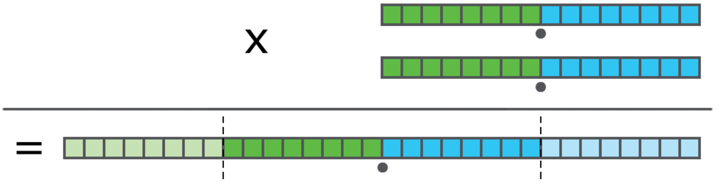
Με βάση όσα προαναφέραμε όμως, η τελευταία εξίσωση μπορεί να γραφτεί ως εξής:

$$F_{n+1} = F_n - (F_n \gg k)$$

Αυτό ήταν! Καταφέραμε να αντικαταστήσουμε τις πολύπλοκες πράξεις που κρύβονται πίσω από μια εκθετική συνάρτηση με μία αφαίρεση και μια ολίσθηση (bit shift). Αυτό που μένει να διευκρινίσουμε είναι ο ρόλος του μεγέθους k. Πολύ απλά, πρόκειται για μια σταθερά της επιλογής μας, η οποία καθορίζει το πόσο γρήγορα θα μειώνεται η εκθετική συνάρτηση. Όσο αυξάνει το k η συνάρτηση μειώνεται με μικρότερο ρυθμό — κι αντίστροφα.

Κάπου εδώ σάς οφείλουμε μερικές εξηγήσεις. Οι πράξεις που κάναμε προηγουμένως, όπως επίσης η αυθαίρετη (και βολική) επιλογή για το γινόμενο των T_c και T_s, συγκαλύπτουν πάρα πολλές υποχωρήσεις στην ακρίβεια του συστήματος. Οι καμπύλες που προκύπτουν επιλύοντας τις εκθετικές

συναρτήσεις με τον τρόπο που παρουσιάσαμε θα είναι *κατά προσέγγιση* εκθετικές και, στην πράξη, θα απαρτίζονται από μικρά ευθύγραμμα τμήματα. Ο λόγος για τον οποίο επιτρέπονται όλες αυτές οι εκπτώσεις στην ακρίβεια είναι απλός: Τα ηχητικά κύματα που παράγει το πρόγραμμά μας συνθέτονται από δείγματα των 8 bits. Με άλλα λόγια, το πρόγραμμα καταλήγει πάντα σε έναν ακέραιο αριθμό από το 0 έως το 255, τον οποίο και τροφοδοτεί στο κύκλωμα PWM. Το γεγονός ότι η τελική έξοδος του προγράμματος είναι τόσο χοντροκομμένη, καθιστά περιττή οποιαδήποτε προσπάθεια βελτίωσης της ακρίβειας.



Εικόνα 6
*Πολλαπλασιάζοντας δύο 16μπιτους αριθμούς προκύπτει ένας 32μπιτος. Για τις ανάγκες του προγράμματός μας θα έπρεπε να κρατήσουμε τα 16 *κεντρικά* bits του αποτελέσματος και να απορρίψουμε τα υπόλοιπα ως overflow και underflow. Για να γίνουν σωστά αυτές οι πράξεις, οι δύο παράγοντες θα πρέπει να μετατραπούν (cast) σε 32μπιτους αριθμούς (long int) και, στη συνέχεια, αφού ολισθήσουμε το αποτέλεσμα (bit shift) κατά 8 bits προς τα δεξιά, να το μετατρέψουμε (cast) σε 16μπιτο (int). Τελικά, στο πρόγραμμά μας αποφασίσαμε να κάνουμε κάτι πολύ πιο απλό...*

Κατασκευαστικές λεπτομέρειες

Πριν προχωρήσουμε στην παρουσίαση του ανανεωμένου κώδικα, θα ήταν καλό να του ρίξετε μια ματιά και να τον συγκρίνετε με την προηγούμενη εκδοχή. Μπορείτε να κατεβάσετε το σχετικό πακέτο από το

`http://bit.ly/dh016ddspianov2`

Όπως θα διαπιστώσετε αμέσως, οι σημαντικότερες αλλαγές εντοπίζονται στη ρουτίνα εξυπηρέτησης του interrupt, στην οποία υπολογίζεται κάθε φορά το νέο δείγμα. Εκεί έχουν προστεθεί οι απαραίτητες πράξεις για τον υπολογισμό των τιμών των εκθετικών συναρτήσεων. Αυτές οι τιμές πολλαπλασιάζονται στη συνέχεια με τα δείγματα του κύματος και με αυτόν τον τρόπο διαμορφώνουν το πλάτος της παραγόμενης ταλάντωσης. Αυτή η περιγραφή όμως παραείναι αφηρημένη. Ας δούμε τι έχει αλλάξει στον κώδικα από κοντά...

Κατά τα γνωστά, ο κώδικας ξεκινά με τις δηλώσεις των μεταβλητών. Εδώ έχουμε προσθέσει τρεις σταθερές, οι οποίες θα χρησιμοποιούνται από τον κώδικα ως «σταθερές χρόνου» για μια εκθετική συνάρτηση. Η πρώτη σταθερά θα χρησιμοποιείται για την αύξουσα εκθετική συνάρτηση (στο ξεκίνημα του volume envelope), ενώ οι άλλες δύο θα χρησιμοποιούνται για τη φθίνουσα εκθετική συνάρτηση (στο τελείωμα του volume envelope). Αναρωτιέστε γιατί ορίζουμε δύο σταθερές για τη φθίνουσα εκθετική συνάρτηση; Όπως αναφέραμε νωρίτερα, θέλουμε το volume envelope να φθίνει ταχύτερα όταν ο χρήστης αφήνει το πλήκτρο, απ' ό,τι όταν το κρατάει πατημένο. Για το σκοπό αυτό, η φθίνουσα εκθετική συνάρτηση θα υπολογίζεται με μια μεγάλη σταθερά χρόνου, όσο το κουμπί είναι πατημένο (ο ρυθμός πτώσης θα είναι μικρός), και με μια μικρότερη, όταν ο χρήστης αφήνει το κουμπί (ο ρυθμός πτώσης θα είναι μεγάλος). Στην ίδια περιοχή του προγράμματος έχουμε ορίσει και τις μεταβλητές που θα διατηρούν τις τιμές των εκθετικών συναρτήσεων. Για τις εν λόγω μεταβλητές χρησιμοποιούμε το union που ορίσαμε στην πρώτη εκδοχή του προγράμματος (my_int), που παρουσιάσαμε στο σχετικό άρθρο του deltaHacker 015. Βλέπετε, οι τιμές των εκθετικών συναρτήσεων πρέπει να μεταβάλλονται όσο πιο ομαλά επιτρέπουν οι μαθηματικές απλοποιήσεις και οι πράξεις στις οποίες έχουμε καταλήξει. Γι' αυτό το λόγο οι τιμές των εκθετικών συναρτήσεων θα αναπαρίστανται με δεκαδικούς αριθ-

μούς, σαν αυτούς που περιγράψαμε στο προηγούμενο άρθρο της σειράς (σταθερής υποδιαστολής 8 προς 8). Ως εκ τούτου, η χρήση του συγκεκριμένου union θα είναι ιδιαίτερα εξυπηρετική, αφού θα μας παρέχει άμεση πρόσβαση στο ακέραιο μέρος των αριθμών (στο περισσότερο σημαντικό byte).

Μία ακόμα προσθήκη στο πρόγραμμα αποτελεί η συνάρτηση init_params(). Αυτή καλείται κάθε φορά που εντοπίζεται ένας πατημένος διακόπτης, αλλά το περιεχόμενό της εκτελείται μόνο σε κάθε νέο πάτημα πλήκτρου. Με άλλα λόγια, αν πατήσουμε ένα κουμπί και το κρατήσουμε πατημένο, το περιεχόμενο της συνάρτησης θα εκτελεστεί μόνο μια φορά: στην αρχή. Σε αυτή τη συνάρτηση αρχικοποιούνται οι εκθετικές συναρτήσεις που σχηματίζουν το περίγραμμα της έντασης. Συγκεκριμένα, η αύξουσα εκθετική συνάρτηση παίρνει την τιμή μηδέν, ενώ η φθίνουσα εκθετική συνάρτηση παίρνει τη μέγιστη τιμή της έντασης. Παρεμπιπτόντως, σαν μέγιστη τιμή για την ένταση επιλέξαμε τον μέγιστο αριθμό που μπορεί να περιγραφεί από 16 bits (65535). Αυτό το κάναμε ώστε οι εκθετικές συναρτήσεις μας, οι τιμές των οποίων αποθηκεύονται σε 16 bits, να εκμεταλλεύονται όλο το διαθέσιμο εύρος τιμών. Η αύξουσα συνάρτηση δηλαδή θα ξεκινά από το 0 και θα φτάνει ως το 65535, ενώ η φθίνουσα θα ακολουθεί την αντίστροφη πορεία.

Ακρωτηριασμός, αριθμητικός

Ένα ακόμα σημείο του προγράμματος που παρουσιάζει ενδιαφέρον, είναι εκείνο στο οποίο πολλαπλασιάζονται οι δύο εκθετικές συναρτήσεις. Η τιμή καθεμίας διατηρείται σε μία 16μπιτη μεταβλητή, την οποία αντιμετωπίζουμε ως δεκαδικό αριθμό με την υποδιαστολή ανάμεσα στο όγδοο και στο ένατο bit. Αυτό σημαίνει ότι το γινόμενό τους θα δώσει έναν 32μπιτο δεκαδικό αριθμό, με την υποδιαστολή ανάμεσα στο δέκατο έκτο και στο δέκατο έβδομο bit. Αμάν! Τι τέρας είναι αυτό, βρε παιδιά; Από αυτό το αποτέλεσμα θα έπρεπε να κρατήσουμε τα 16 *μεσαία* bits, αντιμετωπίζοντας τα υπόλοιπα ως... περίσσευμα (overflow και underflow). Αν αυτή η περιγραφή σάς μπέρδεψε, είναι απόλυτα φυσιολογικό και δεν χρειάζεται να ανησυχείτε. Οι πράξεις που περιγράψαμε άλλωστε είναι σχετικά πολύπλοκες και η σωστή τους εκτέλεση προϋποθέτει μεγαλύτερη προσοχή απ' όσο υποψιάζεται κανείς. Κάπως έτσι αποφασίσαμε να καταφύγουμε πάλι σε μια βάρβαρη απλοποίηση. Συγκεκριμένα, αποφασίσαμε να πολλαπλασιάσουμε μόνο τα ακέραια μέρη των τιμών των εκθετικών συναρτήσεων. Με αυτόν τον τρόπο, ο πολλαπλασιασμός των δύο συναρτήσεων ανάγεται στον πολλαπλασιασμό δύο απλών 8μπιτων αριθμών. Το αποτέλεσμα από αυτή την πράξη είναι ένας 16μπιτος αριθμός, τον οποίο αντιμετωπίζουμε πάλι σαν δεκαδικό αριθμό σταθερής υποδιαστολής! Όπως αντιλαμβάνεστε, όλα αυτά επιταχύνουν την εκτέλεση του προγράμματος, αλλά την ίδια στιγμή αποτελούν και μια μεγάλη αυθαιρεσία — αδικαιολόγητη από μαθηματικής σκοπιάς. Ωστόσο, το γεγονός ότι *όλες* οι πράξεις του προγράμματος καταλήγουν σε ένα 8μπιτο αποτέλεσμα, αφήνει τα περιθώρια για πολλές και χοντροκομμένες απλοποιήσεις.

Αν νομίζετε ότι μετά από όλα αυτά τα τερτίπια σακατέψαμε τον ήχο του πιάνου μας, κάνετε λάθος. Μια μικρή δοκιμή θα σας πείσει αμέσως για το αντίθετο. Μπορεί τα παραγόμενα ηχητικά κύματα να είναι αρμονικά και να παρουσιάζουν σποραδικές ασυνέχειες, αλλά θα δυσκολευτείτε να τις αντιληφθείτε. Ο ήχος του πιάνου θα ακούγεται αρκετά πιο ρεαλιστικός!



Skill: Intermediate

Tags: redirection, pipelining, job control, cron, screen, IPC, inter-process communication

Κυριαρχήστε_στο_command_line:/

part 3/4

Αξιώστε ΠΛΗΡΗ ΕΛΕΓΧΟ!

Έφτασε η ώρα να ανακαλύψετε γιατί οι προχωρημένοι χρήστες του Linux λατρεύουν τη γραμμή εντολών. Μη φαντάζεστε κάποιο φοβερό και τρομερό εργαλείο, το οποίο μπορεί να κάνει τα πάντα. Το μυστικό κρύβεται στις υποδομές της ίδιας της γραμμής εντολών, οι οποίες απελευθερώνουν τη δημιουργική φαντασία του χρήστη!

by Spir@l Evolut10n

Στα δύο πρώτα άρθρα της σειράς μας, στο deltaHacker 015, γνωρίσαμε αρκετές λεπτομέρειες αναφορικά με τη διάρθρωση του συστήματος, καθώς και το χειρισμό του. Ταυτόχρονα, αποκτήσαμε μια στοιχειώδη τουλάχιστον εξοικείωση με διάφορα μικρά προγράμματα. Αν πιστεύετε ότι όσα μάθαμε έχουν μικρή πρακτική αξία, κάνετε τεράστιο λάθος. Εμείς θα σας προτείναμε να ξετινάξετε καθένα από τα μικρά εργαλεία που αναφέρθηκαν και θα αναφερθούν — ακόμη και το φαινομενικά πιο ασήμαντο παράδειγμα. Καλό θα ήταν, επίσης, να μελετήσετε τα σχετικά κείμενα βοήθειας, χρησιμοποιώντας το man. Βλέπετε, σε αυτό το άρθρο θα μάθουμε να συνδυάζουμε τις δυνάμεις κάθε επιμέρους προγράμματος, ώστε να δημιουργούμε τα δικά μας, σύνθετα και πανίσχυρα εργαλεία! Όπως και με τα τουβλάκια Lego, μοναδικό εμπόδιο στο τι θα καταφέρνουμε θα αποτελεί η φαντασία μας!

Ανακατεύθυνση

Για την είσοδο (δεδομένα προς επεξεργασία) και την έξοδο (δεδομένα που προκύπτουν από την επεξεργασία) των προγραμμάτων στο Linux, χρησιμοποιούνται οι λεγόμενες ροές χαρακτήρων. Πρόκειται για τα λεγόμενα character streams, για τα οποία όσοι έχουν ασχοληθεί με τη γλώσσα C είναι βέβαιο ότι έχουν ακούσει. Όμως τα character streams είναι ο μηχανισμός που χρησιμοποιείται από το σύστημα και για το χειρισμό των αρχείων. Ακριβώς γι’ αυτό, τα δεδομένα εισόδου κι εξόδου ενός προγράμματος αντιμετωπίζονται από το Linux ως ένα είδος αρχείων. Ο ενιαίος τρόπος αντιμετώπισης αρχείων και δεδομένων που δέχεται ή παράγει ένα πρόγραμμα, επιτρέπει την υλοποίηση μιας σπουδαίας δυνατότητας. Ο λόγος γίνεται για τη λεγόμενη ανακατεύθυνση (redirection). Πριν επεκταθούμε στη χρησιμότητά της, όμως, πρέπει να αναφέρουμε μερικά ακόμα στοιχεία. Η καθιερωμένη ροή που χρησιμοποιείται για την είσοδο δεδομένων στα προγράμματα θεωρείται το πληκτρολόγιο κι ονομάζεται standard input. Αντίστοιχα, η καθιερωμένη ροή για την έξοδο δεδομένων από τα προγράμματα είναι η οθόνη κι ονομάζεται standard output. Τέλος, υπάρχει και μία ακόμα ροή που χρησιμοποιείται ως έξοδος κι ονομάζεται standard error — νομίζουμε ότι καταλαβαίνετε ποια μηνύματα πηγαίνουν εκεί. Συνήθως, η εν λόγω ροή κατευθύνει τα δεδομένα στην οθόνη, όπως συμβαίνει και με την standard output. Ας επιστρέψουμε τώρα στην έννοια της ανακατεύθυνσης.

Η ανακατεύθυνση παρέχει τη δυνατότητα αποστολής οποιασδήποτε εκ των καθιερωμένων ροών εισόδου κι εξόδου, σε κάποιο αρχείο. Έτσι, μπορούμε για παράδειγμα να στείλουμε την έξοδο από την εκτέλεση ενός προγράμματος σε ένα αρχείο ή να τροφοδοτήσουμε την είσοδο ενός προγράμματος με τα περιεχόμενα ενός αρχείου. Για την ανακατεύθυνση της εξόδου χρησιμοποιείται ο χαρακτήρας >, ενώ για την είσοδο ο <. Βέβαια, πρέπει να σημειώσουμε ότι η ανακατεύθυνση της εισόδου δεν χρησιμοποιείται συχνά. Βλέπετε, τα περισσότερα προγράμματα δέχονται ως παράμετρο κάποιο αρχείο, από το οποίο αντλούν τα προς επεξεργασία δεδομένα. Επομένως, η ανάγκη για ανακατεύθυνση της εισόδου προκύπτει σπάνια. Πριν περάσουμε σε μερικά χαρακτηριστικά παραδείγματα, πρέπει να αναφέρουμε κάτι ακόμα. Μερικές φορές, όταν ανακατευθύνουμε την έξοδο ενός προγράμματος, απαιτείται να προσδιορίσουμε το είδος της εξόδου (standard ή error) που θέλουμε να στείλουμε στον επιθυμητό προορισμό. Για το λόγο αυτό, επιστρατεύονται δύο αριθμητικά ψηφία, που τοποθετούνται ακριβώς πριν από τον χαρακτήρα >. Έτσι, ο συνδυασμός 1> σημαίνει την ανακατεύθυνση του standard output, ενώ ο

συνδυασμός 2> σημαίνει την ανακατεύθυνση του standard error. Βέβαια, όταν κάποιος θέλει να αναφερθεί στο standard output, μπορεί απλά να χρησιμοποιήσει το > κι όχι το 1>. Ας δούμε τώρα τα παραδείγματα που υποσχεθήκαμε.

`cat my_notes > backup`

Τα περιεχόμενα του αρχείου my_notes δεν θα τυπωθούν στην οθόνη, αλλά θα πάνε στο αρχείο backup.

`ls -lh > dir_listing`

Η λίστα με τα αρχεία του τρέχοντος καταλόγου, αποθηκεύεται στο αρχείο dir_listing.

`some_program 2> error_log`

Σε αυτό το παράδειγμα, ανακατευθύνεται μόνο η ροή standard error. Έτσι, όλα τα φυσιολογικά μηνύματα και τα δεδομένα από την εκτέλεση του προγράμματος some_program θα εμφανιστούν κανονικά στην οθόνη. Ωστόσο, τα τυχόν μηνύματα λάθους που ενδέχεται να προκύψουν από τη λειτουργία του προγράμματος, θα πάνε στο αρχείο error_log.

`some_program 2> /dev/null`

Αυτή τη φορά, τα μηνύματα λάθους που παράγονται από το some_program θα εξαφανιστούν και δεν θα τα βρούμε ποτέ και πουθενά. Το device file ονόματι null αποτελεί την καταβόθρα του συστήματος ;) Οτιδήποτε στείλουμε εκεί, χάνεται για πάντα!

Μετά από τα παραπάνω παραδείγματα, πρέπει να σταθούμε σε μια σημαντική λεπτομέρεια. Αν τα αρχεία προς τα οποία γίνεται η ανακατεύθυνση δεν υπάρχουν, το σύστημα θα φροντίσει να τα δημιουργήσει αυτόματα. Αν όμως υπάρχουν, τότε τα περιεχόμενά τους θα αντικατασταθούν και στη θέση τους θα τοποθετηθούν τα νέα! Για να κατανοήσουμε τον κίνδυνο που κρύβεται εδώ, θεωρείστε ότι έχουμε ένα μεγάλο αρχείο ονόματι moan (από το «mother of all notes»), στο οποίο κρατάμε τις σημειώσεις από τη δουλειά ενός έτους. Υποθέστε τώρα ότι δίνουμε το ακόλουθο:

```
cat some_notes > moan
```

Αυτό ήταν! Μπορούμε να κοπανήσουμε το κεφάλι μας στον τοίχο, like, ελεύθερα! Το πολύτιμο περιεχόμενό του moan αντικαταστάθηκε από το περιεχόμενο του αρχείου some_notes :S Για την αποφυγή τέτοιων περιπτώσεων, το κέλυφος προσφέρει μια ενδιαφέρουσα δυνατότητα, η οποία μπορεί να μας γλυτώσει από πολλούς μπελάδες. Αναφερόμαστε στη λεγόμενη διπλή ανακατεύθυνση. Πρακτικά, η διπλή ανακατεύθυνση συμπεριφέρεται όπως και η απλή, με τη σημαντική διαφορά όμως ότι δεν αντικαθιστά τα παλιά δεδομένα των αρχείων, αλλά προσθέτει στο τέλος τους τα νέα. Η διπλή ανακατεύθυνση συμβολίζεται με τους χαρακτήρες >> και για τη χρήση της αρκεί να ακολουθήσουμε την ίδια σύνταξη. Επομένως, στην περίπτωση του τελευταίου παραδείγματος, για να μη χάσουμε τα δεδομένα του moan θα έπρεπε να γράψουμε κάτι τέτοιο:

```
cat some_notes >> moan
```

Διασωλήνωση

Η ανακατεύθυνση είναι ιδιαίτερα εξυπηρετική όποτε θέλουμε να κρατήσουμε την έξοδο κάποιων προγραμμάτων σε αρχεία ή όταν απλά θέλουμε

να την εξαφανίσουμε (βλέπε /dev/null). Άραγε, γίνεται να αξιοποιηθεί η έξοδος ενός προγράμματος πιο δημιουργικά; Η απάντηση είναι καταφατική και η λύση δίνεται από τη λεγόμενη διασωλήνωση (pipelining). Στην ουσία, πρόκειται για έναν μηχανισμό του λειτουργικού συστήματος που χρησιμοποιείται για την επικοινωνία μεταξύ των εφαρμογών (IPC, Inter-Process Communication). Με τη βοήθεια της διασωλήνωσης, λοιπόν, μπορούμε να πάρουμε την έξοδο ενός προγράμματος και να τη χρησιμοποιήσουμε ως είσοδο για κάποιο άλλο. Έτσι, μπορούμε κατά κάποιον τρόπο να συνδέσουμε μεταξύ τους πολλά απλά εργαλεία και να σχηματίσουμε κάποιο που θα συνδυάζει τις ικανότητες των επιμέρους! Ο χαρακτήρας που χρησιμοποιείται για τη διασωλήνωση είναι ο | (pipe). Ας δούμε μερικά παραδείγματα.

```
ls -lh | wc -l
```

Το πρόγραμμα wc (word count) μπορεί, μεταξύ άλλων, να καταμετρά λέξεις, γραμμές, χαρακτήρες, κ.λπ. Με την παράμετρο -l μετράει τις γραμμές. Στο συγκεκριμένο παράδειγμα, η έξοδος από την εκτέλεση του ls -lh θα δοθεί ως είσοδος στο πρόγραμμα wc. Έτσι, αυτό που θα πάρουμε από την εκτέλεση του παραπάνω είναι το πλήθος αρχείων του τρέχοντος καταλόγου!

```
ls -lR | less
```

Το προγραμματάκι less χρησιμοποιείται για την προβολή των περιεχομένων ενός αρχείου. Σε αντίθεση με το cat, όμως, δεν αραδιάζει όλες τις γραμμές του αρχείου αλλά εμφανίζει μόνον όσες χωράνε στην οθόνη. Ταυτόχρονα, επιτρέπει στον χρήστη να χρησιμοποιήσει τα βελάκια για να μετακινηθεί προς τα πάνω ή προς κάτω, ώστε να μελετήσει τα περιεχόμενα του αρχείου με άνεση. Για να τερματιστεί η λειτουργία του less, αρκεί ένα πάτημα στο πλήκτρο [Q]. Όπως αντιλαμβάνεστε, στέλνοντας την έξοδο του ls στο less (με τη βοήθεια της διασωλήνωσης) μπορούμε να εξετάσουμε την ενδεχομένως μακροσκελή λίστα αρχείων ενός καταλόγου με την ησυχία μας.

```
ps -aux | grep pvar > pvar_processes
```

Το προγραμματάκι ps εμφανίζει πληροφορίες σχετικές με τις διεργασίες που εκτελούνται στο σύστημα. Με τις παραμέτρους aux εμφανίζει όλες τις εκτελούμενες διεργασίες και, μεταξύ άλλων, δίπλα από την καθεμία εμφανίζει το όνομα του αντίστοιχου χρήστη. Όσο για το grep, αυτό αποτελεί ένα εξελιγμένο και πανίσχυρο φίλτρο. Στην είσοδό του δέχεται τα περιεχόμενα ενός αρχείου και στην έξοδό του εμφανίζει τις γραμμές που ικανοποιούν το κριτήριο αναζήτησης που έχουμε προσδιορίσει. Στο παράδειγμα που εξετάζουμε, το grep θα επιστρέψει όσες γραμμές περιέχουν τη λέξη pvar. Οπότε, τροφοδοτώντας το με την έξοδο του ps θα λάβουμε μια λίστα με τις διεργασίες που εκτελεί ο χρήστης pvar. Τέλος, με τη βοήθεια της ανακατεύθυνσης οι σχετικές πληροφορίες θα τοποθετηθούν στο αρχείο ονόματι pvar_processes.

Διαδοχική εκτέλεση προγραμμάτων

Ορισμένες φορές θα θελήσουμε να εκτελέσουμε πολλά προγράμματα με μία κίνηση, χωρίς κατ' ανάγκη να συνδυάσουμε τις λειτουργίες τους. Σ' αυτές τις περιπτώσεις πρέπει να έχουμε υπόψη μας τις μεθόδους που προσφέρει το κέλυφος BASH, για την αλυσιδωτή εκτέλεση πολλών προγραμμάτων. Οι εν λόγω μέθοδοι είναι τρεις και το συντακτικό τους είναι

εξαιρετικά απλό: Το ένα πρόγραμμα τοποθετείται δίπλα από το άλλο, ενώ ανάμεσά τους βάζουμε, ως διαχωριστικά, τους κατάλληλους χαρακτήρες. Πάντως, η συμπεριφορά που παρουσιάζει κάθε μέθοδος είναι διαφορετική. Για την ακρίβεια, διαφέρουν ως προς τη λογική με την οποία αποφασίζεται το αν θα ολοκληρωθεί η διαδοχική εκτέλεση των προγραμμάτων ή όχι. Ας δούμε ένα παράδειγμα κάθε μεθόδου και το τοπίο θα ξεκαθαρίσει αμέσως.

program1 && program2

Χρησιμοποιώντας τους χαρακτήρες &&, το κέλυφος θα προχωρήσει στην εκτέλεση του program2 μόνον αν η εκτέλεση του program1 ολοκληρωθεί με επιτυχία.

program1 || program2

Με τους χαρακτήρες ||, το κέλυφος θα προχωρήσει στην εκτέλεση του program2 μόνον εφόσον η εκτέλεση του program1 απέτυχε.

program1 ; program2

Η χρήση του ελληνικού ερωτηματικού θα πρέπει να σας είναι γνώριμη από την ενασχόληση με τα aliases. Χρησιμοποιώντας το ελληνικό ερωτηματικό, το κέλυφος θα προχωρήσει στην εκτέλεση του program2 ανεξάρτητα από το αν πέτυχε η εκτέλεση του program1 ή όχι.

Τέλος, πρέπει να αναφέρουμε ότι με τη βοήθεια των παραπάνω μεθόδων μπορούμε να εκτελέσουμε και παραπάνω από δύο προγράμματα. Για παράδειγμα, θα μπορούσαμε να δώσουμε κάτι τέτοιο:

```
program1 && program2 && program3 && ...
```

Επίσης, μπορούμε αν θέλουμε να συνδυάσουμε και τις τρεις μεθόδους, πληκτρολογώντας κάτι τέτοιο:

```
program1 && program2 || program3 && program4 ; program5 && …
```

Βέβαια, αν αποφασίσουμε να εκτελέσουμε κάτι τόσο σύνθετο θα πρέπει να έχουμε κατά νου ότι η τελική έκβαση δεν είναι εύκολο να προβλεφθεί. Στο συγκεκριμένο παράδειγμα, το program2 θα εκτελεστεί αν προηγουμένως εκτελεστεί με επιτυχία το program1. Ακολούθως, αν το program1 ή το program2 απέτυχε να εκτελεστεί (ή αν απέτυχαν και τα δύο), το κέλυφος θα προσπαθήσει να εκτελέσει το program3. Αν τα καταφέρει, θα προχωρήσει στην εκτέλεση του program4. Μετά από αυτά, ανεξάρτητα από το αν και ποιο πρόγραμμα εκτελέστηκε μέχρι εκείνη τη στιγμή, το κέλυφος θα εκτελέσει το program5… Μπερδευτήκατε; Όταν προσπαθούμε να αναλύσουμε τόσο σύνθετες εκφράσεις, πρέπει να έχουμε υπόψη ότι οι χαρακτήρες && έχουν προτεραιότητα έναντι των ||. Τέλος, μπορούμε να φανταζόμαστε το ελληνικό ερωτηματικό σαν την τελεία που χωρίζει δύο προτάσεις.

Διαχείριση εργασιών

Όπως κάθε σύγχρονο λειτουργικό, έτσι και το Linux είναι πολυδιεργαστικό σύστημα. Με απλά λόγια, μπορεί να κάνει πολλά πράγματα ταυτόχρονα. Όμως πόσο ωφέλιμη θα ήταν αυτή η δυνατότητα, αν κάθε διεργασία καταλάμβανε μια ολόκληρη κονσόλα; Θα μπορούσαμε βέβαια να λύσουμε το πρόβλημα με το να συνδεθούμε σε πολλές κονσόλες ταυτόχρονα. Όμως αυτή η πρακτική θα σπαταλούσε άδικα πόρους του συστήματος και

φυσικά δεν θα ήταν καθόλου κομψή. Ευτυχώς, το συγκεκριμένο ζήτημα μπορεί να αντιμετωπιστεί με πολύ πιο έξυπνους κι αποδοτικούς τρόπους.

Ας υποθέσουμε ότι θέλουμε να εκτελέσουμε το πρόγραμμα mpg123, ώστε να αποκωδικοποιήσουμε ένα αρχείο MP3 και να στείλουμε το αποτέλεσμα σε ένα αρχείο[1]. Με αυτόν τον τρόπο, θα μετατρέπαμε το συγκεκριμένο αρχείο MP3 σε WAV. Όμως, κάνοντας κάτι τέτοιο η κονσόλα θα παραμείνει κατειλημμένη έως ότου ολοκληρωθεί η επεξεργασία του αρχείου MP3. Υπάρχει άραγε τρόπος το κέλυφος να παραμείνει διαθέσιμο, για την ταυτόχρονη εκτέλεση άλλων προγραμμάτων; Για να πετύχουμε το σκοπό μας, αρκεί να χρησιμοποιήσουμε τον ειδικό χαρακτήρα &. Συγκεκριμένα, κατά την εκτέλεση του mpg123 κι αφού δώσουμε τις κατάλληλες παραμέτρους, στο τέλος της γραμμής θα πρέπει να εισαγάγουμε ένα &. Εν ολίγοις, θα πρέπει να εκτελέσουμε κάτι τέτοιο:

```
mpg123 -s  my_song.mp3 > my_song.wav &
```

Όταν χρησιμοποιούμε το χαρακτήρα &, το σύστημα τοποθετεί την εκτέλεση του εκάστοτε προγράμματος στο λεγόμενο υπόβαθρο (background). Έτσι, η κονσόλα παραμένει διαθέσιμη για την εκτέλεση οποιουδήποτε άλλου προγράμματος θέλουμε. Μια εναλλακτική μέθοδος για να πετύχουμε το ίδιο αποτέλεσμα είναι να εκτελέσουμε το επιθυμητό πρόγραμμα κανονικά κι όταν ξεκινήσει να πατήσουμε το συνδυασμό πλήκτρων [CTRL+Z]. Με αυτόν τον τρόπο, το συγκεκριμένο πρόγραμμα θα πάψει προσωρινά να εκτελείται (suspend) κι εμείς θα επιστρέψουμε στην κονσόλα. Για να συνεχιστεί η λειτουργία του προγράμματος χωρίς να δεσμευτεί η κονσόλα, αρκεί να τρέξουμε το πρόγραμμα bg (από το background). Διαφορετικά, αν αλλάζαμε γνώμη κι επιθυμούσαμε την κανονική εκτέλεση του προγράμματος (επιτρέποντάς του να καταλάβει την κονσόλα), θα αρκούσε να τρέξουμε το πρόγραμμα fg (από το foreground). Χρησιμοποιώντας αυτή τη μέθοδο, σύντομα θα καταλήξουμε σε μία κατάσταση κατά την οποία εκτελούνται πολλά προγράμματα στο υπόβαθρο. Σε αυτή την περίπτωση, είναι πολύ πιθανό να χρειαστούμε τη βοήθεια του προγράμματος jobs. Εκτελώντας το, εμφανίζεται μια λίστα με όλα τα προγράμματα που έχουν τοποθετηθεί στο υπόβαθρο. Μάλιστα, σε κάθε γραμμή και πριν από το όνομα του εκάστοτε προγράμματος, εμφανίζεται κι ένας αριθμός. Για να επαναφέρουμε στο προσκήνιο τη λειτουργία κάποιου προγράμματος, αρκεί να δώσουμε τον αντίστοιχο αριθμό ως παράμετρο στο fg. Κάντε μερικές δοκιμές και δείτε.

Καθολικός έλεγχος

Ακόμη και στα πιο στιβαρά λειτουργικά συστήματα, το φαινόμενο του κρεμάσματος μιας εφαρμογής δεν εκλείπει. Άλλωστε, τις περισσότερες φορές αυτό οφείλεται σε ζητήματα που σχετίζονται με την εκάστοτε εφαρμογή κι όχι με το σύστημα. Σε αυτές τις περιπτώσεις, για τη διαχείριση των εργασιών απαιτούνται πιο δραστικά μέτρα. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε το πρόγραμμα killall. Δίνοντας το όνομα ενός προγράμματος σαν παράμετρο στο killall, το σύστημα θα τερματίσει κάθε στιγμιότυπο του συγκεκριμένου προγράμματος. Όμως αυτό ενδέχεται να μην είναι επιθυμητό. Αν για παράδειγμα είχαμε εκτελέσει δύο φορές το πρόγραμμα mpg123 και το ένα στιγμιότυπο είχε κολλήσει, θα ήταν ανόητο να τερματίσουμε και τα δύο. Κάπου εδώ παρουσιάζεται η αναγκαιότητα ενός προγράμματος σαν το ps (από το process tree). Με τη βοήθειά του,

1. Το εν λόγω πρόγραμμα δεν εγκαθίσταται εξ ορισμού στο Ubuntu. Για τις άλλες διανομές δεν θυμόμαστε ;) Σε κάθε περίπτωση, αν αποφασίσετε να παίξετε μαζί του φροντίστε προηγουμένως να το εγκαταστήσετε, από τον διαχειριστή εφαρμογών της εκάστοτε διανομής.

όπως αναφέραμε και νωρίτερα, μπορούμε να δούμε όλες τις διεργασίες που εκτελούνται, καθώς και πολλές σχετικές πληροφορίες. Εκτελώντας το ps με παράμετρο το ux, εμφανίζονται αποκλειστικά οι εφαρμογές που εκτελούνται από το λογαριασμό μας. Έτσι, δεν έχουμε παρά να εντοπίσουμε το στιγμιότυπο του mpg123 που έχει κολλήσει και να σημειώσουμε το PID (Process ID) που του έχει αντιστοιχίσει το σύστημα. Στη συνέχεια, αρκεί να εκτελέσουμε το πρόγραμμα kill και να του δώσουμε σαν παράμετρο το συγκεκριμένο PID. Παρεμπιπτόντως, σε περιπτώσεις που η κολλημένη εφαρμογή δεν τερματίζεται με αυτόν τον τρόπο, μπορούμε να της φερθούμε πιο σκληρά! Αφού πληροφορηθούμε το αντίστοιχο PID, μπορούμε να δώσουμε κάτι τέτοιο:

```
kill -9  αντίστοιχο_PID
```

(Σ.τ.Ε. Κλαίω!)

Προγραμματισμός εργασιών

Τώρα πια έχουμε μάθει αρκετούς τρόπους για να σταματήσουμε ένα πρόγραμμα ή απλά να το στείλουμε στο υπόβαθρο. Αντίθετα, για την εκκίνηση ενός προγράμματος γνωρίζουμε μόνο έναν τρόπο: Την εκτέλεσή του από τη γραμμή εντολών. Φυσικά, η χειροκίνητη εκτέλεση ενός προγράμματος επαρκεί για τις περισσότερες καθημερινές ασχολίες. Ωστόσο, καθώς αυξάνει η εξοικείωσή μας με το Linux και τις δυνατότητές του, θα προκύψει μια νέα ανάγκη: Αναφερόμαστε σε εκείνη της αυτόματης εκτέλεσης προγραμμάτων, σε προκαθορισμένες χρονικές στιγμές. Για το σκοπό αυτό, το σύστημα παρέχει μια υπηρεσία που επιτρέπει στους χρήστες να προγραμματίζουν με ικανοποιητικότατη ακρίβεια την εκτέλεση οποιασδήποτε εφαρμογής. Προκειμένου να εκμεταλλευτούμε αυτή την υπηρεσία, θα πρέπει να χρησιμοποιήσουμε το πρόγραμμα crontab. Συγκεκριμένα, θα ξεκινήσουμε πληκτρολογώντας

```
crontab –e
```

Με αυτόν τον τρόπο θα εκτελεστεί ο default text editor του συστήματος. Στο Ubuntu, πάντως, την πρώτη φορά που θα εκτελεστεί το crontab δεν θα ξεκινήσει αυτόματα κάποιος editor, αλλά θα ερωτηθούμε για τον ποιον προτιμάμε. Αν σας ενδιαφέρει, εμείς αγαπάμε το nano[2] :) Όταν ανοίξει ο text editor, δεν θα δούμε τίποτε απολύτως. Αυτό είναι φυσιολογικό —ειδικά για την πρώτη φορά που θα εκτελέσουμε το crontab— και δεν πρέπει να μας ανησυχήσει. Εκεί, για κάθε εφαρμογή που θέλουμε να αυτοματοποιήσουμε, θα πρέπει να εισαγάγουμε μια γραμμή της ακόλουθης μορφής:

```
j   k    l    m    n    όνομα_εφαρμογής
```

Το j αναφέρεται στα λεπτά (0-59), το k στην ώρα (0-23), το l στην ημέρα του μήνα (1-31), το m στον μήνα (1-12) και το n στην ημέρα της εβδομάδας (0-6). Σημειώστε ότι για κάθε πρόγραμμα δεν αρκεί μόνο το όνομά του, αλλά και η θέση του στους καταλόγους του συστήματος. Για το σκοπό αυτό, πριν εκτελέσουμε το crontab –e, καλό θα ήταν να τρέξουμε το which με παράμετρο το όνομα του επιθυμητού προγράμματος, ώστε να πληροφορηθούμε τη θέση του. Επίσης, όταν κάποιο από τα μεγέθη που προσδιορίζουν το χρόνο εκτέλεσης δεν μας ενδιαφέρει, θα πρέπει να τοποθετήσουμε στη θέση του έναν αστερίσκο. Αφού βάλουμε όσες γραμμές

2. Οι δήθεν Linuxαδες θα γελάσουν με αυτή την επιλογή μας. Εμείς όμως γουστάρουμε να κάνουμε τη δουλειά μας εύκολα και γρήγορα και, πάνω απ' όλα, χωρίς να πουλάμε μούρη. Αυτοί από τη μεριά τους μπορούν να κοπανάνε το κεφάλι τους στον τοίχο. Όσο θέλουν.

θέλουμε, αρκεί να πατήσουμε το συνδυασμό πλήκτρων [CTRL+O] για να αποθηκευτούν οι προσθήκες μας[3]. Ακολούθως, δεν έχουμε παρά να πατήσουμε το συνδυασμό πλήκτρων [CTRL+X] για να βγούμε από τον editor[4]. Από εκείνη τη στιγμή και μετά, τα προγράμματα που εισηγάγαμε προηγουμένως θα αρχίσουν να εκτελούνται αυτόματα, στους χρόνους που προσδιορίσαμε. Αν σας μπέρδεψε η περιγραφή που μόλις κάναμε, δείτε τα ακόλουθα παραδείγματα και θα καταλάβετε αμέσως πόσο απλό και ισχυρό είναι το crontab.

```
30   23       *       *       0       /usr/bin/my_program
```

Το my_program θα εκτελείται κάθε Κυριακή βράδυ, στις 23:30.

```
*/30 *        *       *       *       /usr/bin/my_program
```

Το my_program θα εκτελείται ανά 30 λεπτά!

```
*    18       *       */3     0       /usr/bin/my_program
```

Το my_program θα εκτελείται ανά τρεις μήνες, στις έξι το απόγευμα κάθε Κυριακής του συγκεκριμένου μήνα.

```
30   8        1       */1     *       /usr/bin/my_program
```

Το my_program θα εκτελείται στις 8:30 το πρωί, κάθε πρώτη του μηνός.

Τέλος, αν θέλουμε να δούμε τις εργασίες που έχουν προγραμματιστεί με το crontab, αρκεί να το εκτελέσουμε με παράμετρο το -l. Εάν πάλι θέλουμε να καταργήσουμε όλες τις προγραμματισμένες εργασίες, αρκεί να το εκτελέσουμε με την παράμετρο -r.

Εφαρμογές στο υπόβαθρο — και με την κονσόλα κλειστή!

Φανταστείτε το εξής σενάριο: Συνδεόμαστε στο νέο μας VPS κι εκτελούμε ένα πρόγραμμα, του οποίου η εργασία απαιτεί αρκετό χρόνο. Μήπως αυτό σημαίνει ότι πρέπει να αφήσουμε το τερματικό ανοιχτό; Όχι τίποτα άλλο, αλλά αυτό θα μας υποχρέωνε να αφήσουμε και τον υπολογιστή αναμμένο. Αυτό το ζήτημα μπορεί να προκύψει με οποιοδήποτε σύστημα Linux, στο οποίο συνηθίζουμε να εργαζόμαστε απομακρυσμένα. Εξάλλου, ακόμη κι όταν εργαζόμαστε τοπικά σε ένα σύστημα, η ιδέα του να αφήσουμε ένα τερματικό ανοιχτό και να φύγουμε —έστω και για λίγο— δεν είναι φρόνιμη. Οποιοσδήποτε αρπάξει την ευκαιρία και καθίσει στο σύστημά μας, θα μπορεί να το χρησιμοποιήσει με τα δικαιώματα του λογαριασμού μας! Για την αντιμετώπιση αυτών των καταστάσεων μπορούμε να στραφούμε στο προγραμματάκι screen. Ας υποθέσουμε, λοιπόν, ότι θέλουμε να τρέξουμε ένα πρόγραμμα του οποίου η εκτέλεση καθυστερεί αρκετά. Αυτό που πρέπει να κάνουμε είναι να τρέξουμε αρχικά το screen. Η οθόνη θα καθαρίσει αμέσως και θα μοιάζει σαν να συνδεθήκαμε μόλις σε μία νέα κονσόλα. Στην πράξη, θα έχει ξεκινήσει ένα νέο screen session, όπως συνηθίζουμε να λέμε. Από εκεί, δεν έχουμε παρά να τρέξουμε το (χρονοβόρο) πρόγραμμα, να πατήσουμε το συνδυασμό πλήκτρων [CTRL+A] και στη συνέχεια να πιέσουμε το πλήκτρο [D]. Με το πάτημα αυτών των πλήκτρων θα πραγματοποιηθεί το λεγόμενο detach. Με απλά λόγια, θα αποσυνδεθούμε από το screen session εντός του οποίου εκτελείται η εφαρμογή. Πλέον θα είμαστε σε θέση να αποσυνδεθούμε από την κονσόλα, γνωρίζοντας ότι το screen session —και κατ' επέκταση η εφαρμογή μας— θα συνεχίσουν να εκτελούνται ανενόχλητα! Αργότερα,

3. Ο συγκεκριμένος συνδυασμός πλήκτρων ισχύει μόνο στην περίπτωση που χρησιμοποιούμε το nano. Αν έχετε επιλέξει κάποιον άλλο editor, αρκεί να πατήσετε τον αντίστοιχο συνδυασμό για αποθήκευση του αρχείου.
4. Προφανώς, το ίδιο ισχύει και γι' αυτόν το συνδυασμό.

όταν συνδεθούμε εκ νέου σε κάποια κονσόλα του συστήματος, θα μπορούμε να συνδεθούμε ξανά στο αποσυνδεδεμένο screen. Για το σκοπό αυτό αρκεί να δώσουμε κάτι τέτοιο:

```
screen -r
```

Μ' αυτόν τον τρόπο το ενεργό screen session θα επιστρέψει στο προσκήνιο (attach) και θα βρεθούμε στο «περιβάλλον» εντός του οποίου είχαμε τρέξει το πρόγραμμα. Πριν εμβαθύνουμε στη χρήση του screen, σημειώστε ότι για τον τερματισμό του screen session θα πρέπει να εκτελέσουμε το exit — ή να πατήσουμε το συνδυασμό πλήκτρων [CTRL+D].

Πολλά προγράμματα σε πολλά screen session

Όπως αναφέραμε, με τη βοήθεια του screen μπορούμε να αφήσουμε το μηχάνημά μας να εκτελεί μια εργασία, χωρίς να είμαστε συνδεδεμένοι σε αυτό. Όμως τι γίνεται αν επιθυμούμε την εκτέλεση πολλών προγραμμάτων; Η πιο απλή λύση είναι να ξεκινήσουμε πολλά screen sessions, τρέχοντας το screen όσες φορές επιθυμούμε. Η σχετική διαδικασία έχει ως εξής: Τρέχουμε το screen, ξεκινάμε την πρώτη εργασία, πατάμε [CTRL+A] και μετά [D] (detach). Μετά ξανατρέχουμε το screen, ξεκινάμε τη δεύτερη εργασία και κάνουμε πάλι detach. Αυτό μπορούμε να το επαναλάβουμε για όσα προγράμματα θέλουμε. Τώρα, όμως, αν δώσουμε screen -r θα εμφανιστεί μία λίστα με όλα τα screen sessions που ξεκινήσαμε προηγουμένως. Για να συνδεθούμε σε κάποιο, δεν έχουμε παρά να εκτελέσουμε screen -r και να δώσουμε σαν παράμετρο το PID (Process ID) του επιθυμητού session. Φυσικά τα PID εμφανίζονται στη λίστα με τα screen sessions, που αναφέραμε προηγουμένως. Με αυτόν τον τρόπο μπορούμε να συνδεθούμε διαδοχικά σε κάθε screen session και να ελέγξουμε την πορεία της αντίστοιχης εργασίας. Σε αυτό το σημείο, αρκετοί από εσάς θα σκεφτείτε ότι η επανασύνδεση σε κάποιο screen session χρησιμοποιώντας το αντίστοιχο PID δεν είναι καθόλου εξυπηρετική. Πράγματι, αν ξεκινήσουμε πέντε-έξι screen sessions, αποκλείεται να θυμόμαστε ποιο πρόγραμμα εκτελείται σε ποιο session. Φυσικά, η διαδοχική σύνδεση στο καθένα, εκτός από χρονοβόρα είναι και εκνευριστική. Για να αποφύγουμε αυτή την κατάσταση, είναι δυνατόν να αποδώσουμε σε κάθε session ένα όνομα. Μετέπειτα, για τη σύνδεσή μας σε αυτό θα αρκεί να χρησιμοποιήσουμε το όνομά του και όχι το PID. Η απόδοση ονόματος σε ένα screen session μπορεί να γίνει με δύο τρόπους. Ο πρώτος είναι να δώσουμε το επιθυμητό όνομα κατά την εκκίνηση του session:

```
screen -S port_scanning
```

Έτσι, θα ξεκινήσει ένα screen session με το όνομα port_scanning. Αργότερα, όταν χρειαστεί να επανασυνδεθούμε σε αυτό, θα αρκεί να εκτελέσουμε το ακόλουθο:

```
screen -r port_scanning
```

Αν ξεχάσουμε να βαφτίσουμε κάποιο screen session κατά την εκκίνησή του, μπορούμε να το βαφτίσουμε αργότερα. Ας υποθέσουμε, λοιπόν, ότι έχουμε ξεκινήσει ένα session και δεν έχουμε κάνει ακόμα detach. Για να το βαφτίσουμε αρκεί να πατήσουμε το συνδυασμό [CTRL+A] και στη συνέχεια να πιέσουμε το πλήκτρο με την άνω-κάτω τελεία [:]. Κατ' αυτό τον τρόπο, στην τελευταία γραμμή του screen session θα εμφανιστεί ο εν λόγω χαρακτήρας και το πρόγραμμα θα περιμένει να εισαγάγουμε κάποια εντολή. Όπως αντιλαμβάνεστε, οι εντολές που μπορούμε να δώσουμε εκεί είναι σχετικά περιορισμένες και πρόκειται για «εσωτερικές» εντολές

του screen: εντολές που αναγνωρίζει μόνο το screen και οι οποίες αφορούν στη λειτουργία του. Τελικά, για να βαφτίσουμε το screen session πρέπει να γράψουμε τη λέξη title κι ακολούθως το επιθυμητό όνομα. Στο τέλος αρκεί να δώσουμε ένα [ENTER]. Απλό δεν ήταν;

Πολλά προγράμματα, στο ίδιο screen

Αυτή τη στιγμή έχουμε λύσει το αρχικό μας πρόβλημα: Μπορούμε να τρέχουμε ένα ή περισσότερα προγράμματα στο σύστημά μας, χωρίς να παραμένουμε υποχρεωτικά συνδεδεμένοι. Η λύση που είδαμε, όμως, δεν είναι η καλύτερη δυνατή. Το να γεμίσουμε το σύστημα με ένα σωρό screen sessions δεν είναι ό,τι πιο έξυπνο θα μπορούσε να κάνει κανείς. Το ιδανικό θα ήταν να τρέξουμε όλες τις εφαρμογές που θέλουμε *μέσα* στο ίδιο screen session. Έ, λοιπόν, το screen μπορεί να το κάνει *και* αυτό! Για την ακρίβεια, προσφέρει μία δυνατότητα που θυμίζει τους web browsers: επιτρέπει τη δημιουργία πολλαπλών καρτελών (tabs). Ας δούμε πώς γίνεται κάτι τέτοιο στην πράξη.

Έστω ότι θέλουμε να τρέξουμε δύο προγράμματα, μέσα στο ίδιο session. Αφού ξεκινήσουμε το screen εκτελούμε το πρώτο πρόγραμμα, πατάμε το συνδυασμό [CTRL+A] και μετά το [C]. Με αυτόν το συνδυασμό πλήκτρων, το screen θα δημιουργήσει μια νέα καρτέλα. Εδώ πρέπει να σημειώσουμε οτι δεν θα εμφανιστεί κανένα ιδιαίτερο μήνυμα ούτε κάποιο σύμβολο που θα φανερώνει τη δημιουργία της καρτέλας. Ωστόσο η οθόνη θα καθαρίσει, σαν να είχαμε ξεκινήσει ένα νέο session. Πλέον, μπορούμε να εκτελέσουμε και το δεύτερο πρόγραμμα. Για τη μεταφορά από τη μία καρτέλα στην άλλη μπορούμε να πατήσουμε το συνδυασμό [CTRL+A] κι ακολούθως **α)** το [P] (από το previous) για να μεταβούμε στην προηγούμενη καρτέλα, **β)** το [N] (από το next) για να μεταβούμε στην επόμενη. Βέβαια, στην περίπτωση που εξετάζουμε υπάρχουν μόνο δύο καρτέλες και η επιλογή του [P] δεν θα έχει καμία διαφορά από την επιλογή του [N]. Οποιαδήποτε στιγμή αποφασίσουμε να κάνουμε detach, μπορούμε να το κάνουμε με το γνωστό τρόπο κι αδιαφορώντας για την καρτέλα στην οποία βρισκόμαστε. Αργότερα, για να ελέγξουμε την κατάσταση των προγραμμάτων μας αρκεί να συνδεθούμε στο screen session και να μετακινηθούμε από καρτέλα σε καρτέλα, με τους συνδυασμούς που αναφέραμε προηγουμένως.

Η συνέχεια στις οθόνες σας!

Σε αυτό το άρθρο καταπιαστήκαμε με πολλά προχωρημένα ζητήματα, καθένα από τα οποία θα μπορούσε να αποτελέσει θέμα για ολόκληρο βιβλίο. Αυτή τη στιγμή λοιπόν δεν έχουμε την πεποίθηση ότι σας είπαμε *όλα* όσα οφείλει να γνωρίζει κανείς για το Linux και τη γραμμή εντολών του. Ελπίζουμε μόνο ότι σας πωρώσαμε αρκετά, ώστε να ασχοληθείτε από μόνοι σας. Ταυτόχρονα, θέλουμε να πιστεύουμε ότι σας δώσαμε όλες τις βασικές γνώσεις, ώστε η περαιτέρω μελέτη σας να προχωρήσει αβασάνιστα. Και τώρα ανοίχτε ένα τερματικό κι αρχίστε τις δοκιμές! Μόνο μην ξεχάσετε να διαβάσετε και το τέταρτο μέρος της σειράς μας, που αρχίζει από τη σελίδα 88 του τεύχους που κρατάτε στα χέρια σας. Θέλουμε να σας δείξουμε κάτι ακόμα... ;)



Είμαστε κοινωνικοί -- ειδικά στον online κόσμο.
Είμαστε αστείοι, ευφυείς και φινετσάτοι -- ειδικά στον online κόσμο.
Είναι βέβαιο ότι έχουμε καβαλήσει το καλάμι -- τόσο στον online, όσο και στον offline κόσμο.
Το διαχειριζόμαστε γενναία, πάντως.
Περάστε από τα στέκια μας να τα πούμε -- ή να μην την πείτε.
twitter.com/deltaHacker
facebook.com/deltaHacker
DeltaHacker
6,980 likes · 333 talking about this
Computers/Technology
Μηνιαίο συνδρομητικό περιοδικό με θεματολογία ethical hacking, security, programming & electronics.
deltaHacker
@deltaHacker

Skill: Intermediate

Tags: picture frame, uboot, minifs, embedded systems

Ωραιότατη κορνίζα, αλλά για αληθινούς άντρες!



by GiannouG

Οι ψηφιακές κορνίζες (picture frames) είναι συσκευές που έκαναν την εμφάνισή τους τα τελευταία χρόνια με σκοπό να αντικαταστήσουν τις παλιές, καλές και *στατικές* κορνίζες που είχαμε συνηθίσει. Οι συσκευές αυτές χρησιμοποιούν μια οθόνη για να προβάλλουν φωτογραφίες. Συνήθως περιέχουν αρκετές επιπλέον λειτουργίες, όπως Bluetooth και WiFi για απομακρυσμένη διαχείριση αλλά κι ανέβασμα φωτογραφιών. Οι ψηφιακές κορνίζες δεν είναι τίποτε άλλο από μικροί υπολογιστές, οι οποίοι συνήθως τρέχουν ένα Linux-based λειτουργικό. Το Linux είναι η ευκολότερη, πιο προσβάσιμη (διαθέσιμο το source code) και πιο φθηνή (δωρεάν) λύση, για αυτό και τη συναντάμε σχεδόν παντού — ακόμη και στις κορνίζες μας. Τι θα γινόταν αν μπορούσαμε να τρέξουμε τη δική μας διανομή Linux, πάνω στην ψηφιακή μας κορνίζα; Σίγουρα το hardware δεν φτιάχτηκε για κάτι τέτοιο και θα συναντήσουμε κάποια εμπόδια, αλλά Γιατί Όχι (TM);

Δυστυχώς για εμάς, δεν μπορούμε να πειράξουμε οποιαδήποτε κορνίζα βρούμε με το «έτσι θέλω». Η κορνίζα θα πρέπει α) να έχει το κατάλληλο hardware και β) να υπάρχει η ανάλογη υποστήριξη και προεργασία από άλλους περίεργους, σαν εμάς.

Λίγους μήνες πριν είχε γίνει *πανικός* γύρω από το Parrot DF3120. Πρόκειται για ένα αρκετά μικρό picture frame, ικανό να τρέξει Linux αλλά και κάποια επιπλέον προγράμματα. Η μικρή αυτή ψηφιακή κορνίζα έχει αρκετά ενδιαφέροντα χαρακτηριστικά όπως Bluetooth, αισθητήρα φωτός αλλά και φυσικά υποστήριξη για κάρτες SD. Το πιο εντυπωσιακό χαρακτηριστικό της συσκευής είναι βέβαια η τιμή: μόλις 10 λίρες Αγγλίας, δηλαδή περίπου 12,50 ευρώ! Δυστυχώς για εμάς, οι συσκευές αυτές είναι τόσο περιζήτητες που δε θα τις βρούμε εύκολα. Εμφανίζονται κατά καιρούς στα eBay και Amazon, συνήθως με τιμές αρκετά υψηλότερες από την αρχική. Με λίγη τύχη και πολλή υπομονή, όμως, θα βρούμε κάποια. Ο γράφων, π.χ., βρήκε και προμηθεύτηκε μία — κι ας την πλήρωσε σχεδόν στη διπλάσια τιμή.

Ενδιαφέρον, για συνέχισε...

Ας δούμε επιγραμματικά το όλο εγχείρημα. Καταφέραμε να έχουμε στα χέρια μας το Parrot DF3120, την ψηφιακή κορνίζα που τρέχει το λειτουργικό του κατασκευαστή κι εμφανίζει ό,τι φωτογραφίες έχουμε στη μνήμη ή φωτογραφίες που της στέλνουμε από το κινητό μας, μέσω Bluetooth. Το λειτουργικό αυτό μας είναι *λιγάκι* άχρηστο. Είναι που δεν μας ενδιαφέρει καθόλου η λειτουργία του ως picture frame αλλά η δυνατότητα του hardware να τρέξει Linux. Όχι, δεν θα το πειράξουμε το αρχικό λειτουργικό, γιατί δεν θέλουμε σε καμία περίπτωση να χαλάσουμε την συσκευή[1]. Απλά θα ενσωματώσουμε ένα κομμάτι κώδικα που θα φορτώνει το Linux από την κάρτα SD, μόνον όταν κρατάμε πατημένο κάποιον συνδυασμό πλήκτρων της συσκευής. Αφήνοντας ακέραιη την ενσωματωμένη μνήμη, είμαστε *σχεδόν* σίγουροι ότι πάντα όλα θα πηγαίνουν καλά και η συσκευή μας θα μπορεί να ξεκινά — έστω και με το λειτουργικό του κατασκευαστή :)

Θα ξεκινήσουμε χτίζοντας το minifs, μια μικρή διανομή Linux για συσκευές σαν αυτή. Θα χρησιμοποιήσουμε επίσης το uboot, έναν bootloader[2] για embedded συστήματα, καθώς και το plftool, ένα εργαλείο για την δημιουργία update files που καταλαβαίνει το picture frame μας. Όλα αυτά θα γίνουν σε περιβάλλον Ubuntu διότι, όπως θα έχετε παρατηρήσει, τέτοιες δουλειές γίνονται μόνο στο Linux :D

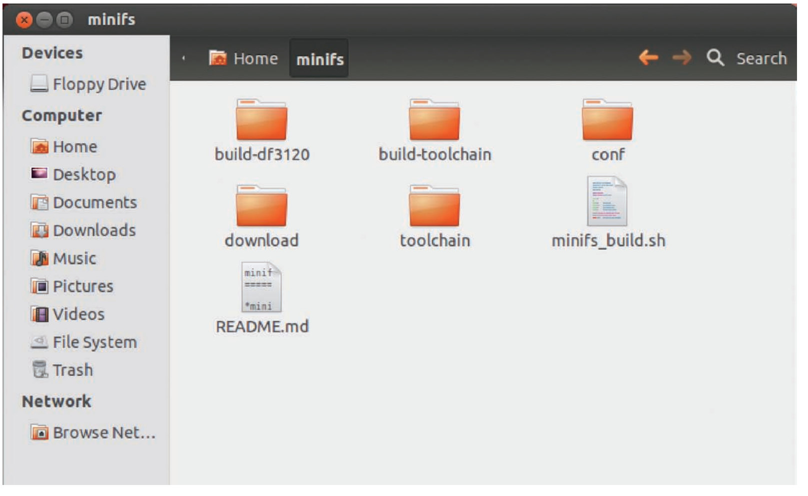
Το picture frame μας είναι αρκετά περιορισμένο, δεδομένου της αρχικής του λειτουργίας και φυσικά της τιμής. Φυσικά, δεν



Ιδού το Parrot DF3120, η μικρή κορνίζα που σε λιγάκι θα τρέχει τη δική μας διανομή Linux!

1. Είμαστε περίεργοι, αλλά δε θέλουμε να χαλάμε τα παιχνίδια μας.
2. Είναι το πρώτο πρόγραμμα που ξεκινά η συσκευή. Για να λέμε την αλήθεια είναι το δεύτερο, μιας κι ο κάθε κατασκευαστής τρέχει πρώτα κάποιο δικό του κώδικα, ώστε να αρχικοποιεί σωστά το hardware.

θα μπορέσει ποτέ να τρέξει κάποια πλήρη διανομή Linux ή κάποια απαιτητικά προγράμματα[3]. Αυτό βέβαια δεν σημαίνει ότι δεν μπορεί να κάνει θαύματα. Η συσκευή είναι χτισμένη πάνω στο Samsung S3C2412. Πρόκειται για ένα SoC[4] που περιέχει 32bit ARM CPU και αρκετά ακόμη καλούδια, τα οποία οι κατασκευαστές της κορνίζας αποφάσισαν να μη χρησιμοποιήσουν. Συνοδεύεται από μόλις 8MB RAM, αλλά κι από 32MB αποθηκευτικού χώρου!



Το minifs μάς βοηθάει στη δημιουργία της διανομής για την κορνίζα μας, αλλά και στις απαραίτητες μεταγλωττίσεις για τα προγράμματα αυτής.

Η δική μας, mini διανομή

Τι ωραία που θα ήταν αν μπορούσαμε να περάσουμε στην κορίζα οποιαδήποτε διανομή θέλουμε, κι αυτή να τρέξει! Από την άλλη βέβαια ευτυχώς που αυτό *δεν* γίνεται, αφού μας αρέσουν τα δύσκολα :) Θα βασιστούμε όμως σε δουλειά άλλων, οι οποίοι ετοίμασαν κάποια scripts για τη δημιουργία μιας μικρής διανομής. Ο κόπος κι ο χρόνος που ξόδεψαν για να σιγουρευτούν πως όλα λειτουργούν σωστά, είναι κάτι που δεν μπορούμε να παραβλέψουμε. Ο λόγος γίνεται για το minifs, ένα σετ από scripts που αναλαμβάνουν τόσο να κατεβάσουν όσο και να ρυθμίσουν τα απαραίτητα προγράμματα για τη λειτουργία του Linux εντός της κορνίζας μας.

Το minifs, λοιπόν, είναι αρκετά BASH scripts που κατεβάζουν κάποια αρχεία και τρέχουν μερικές προκαθορισμένες εντολές κι εργασίες. Μπορούμε πολύ εύκολα να ρυθμίσουμε το minifs όπως εμείς θέλουμε, προσθέτοντας ή αφαιρώντας αρχεία ρυθμίσεων στους φακέλους του. Για αρχή θα βασιστούμε στις προκαθορισμένες ρυθμίσεις, τουλάχιστον μέχρι να βεβαιωθούμε ότι όλα πάνε καλά. Εντελώς τυχαία, ο δημιουργός του minifs είναι από εκείνους που ασχολήθηκαν ειδικά και με το Linux στο DF3120, έτσι το minifs υποστηρίζει εξ αρχής την κορνίζα μας.

Ας ξεκινήσουμε πρώτα ετοιμάζοντας το έδαφος. Όπως είπαμε και πριν, βασιζόμαστε στο Ubuntu, κατά προτίμηση στην έκδοση 12.10. Για το άρθρο χρησιμοποιήσαμε μια καθαρή και φρέσκια εγκατάσταση, σε εικονική μηχανή (Virtual Machine, VM). Τα πακέτα που θα εγκαταστήσουμε χρειάζονται για τη μεταγλώττιση των απαραίτητων προγραμμάτων κι εργαλείων. Αν κάποιο πακέτο λείπει, το minifs θα διακόψει την λειτουργία του

3. Για αυτά υπάρχει το Raspberry Pi ;)
4. System On a Chip, δηλαδή όλα τα βασικά μέρη ενός υπολογιστή μέσα σε ένα chip.

και θα μας ζητήσει να το εγκαταστήσουμε. Με δικαιώματα υπερχρήστη (δώστε sudo su), σε μια κονσόλα τερματικού πληκτρολογούμε:

```
# apt-get install build-essentials git libelf-dev libncurses-dev
lzop curl subversion cvs genext2fs texinfo gawk bison flex gperf
libtool autoconf libglib2.0-dev gettext
```

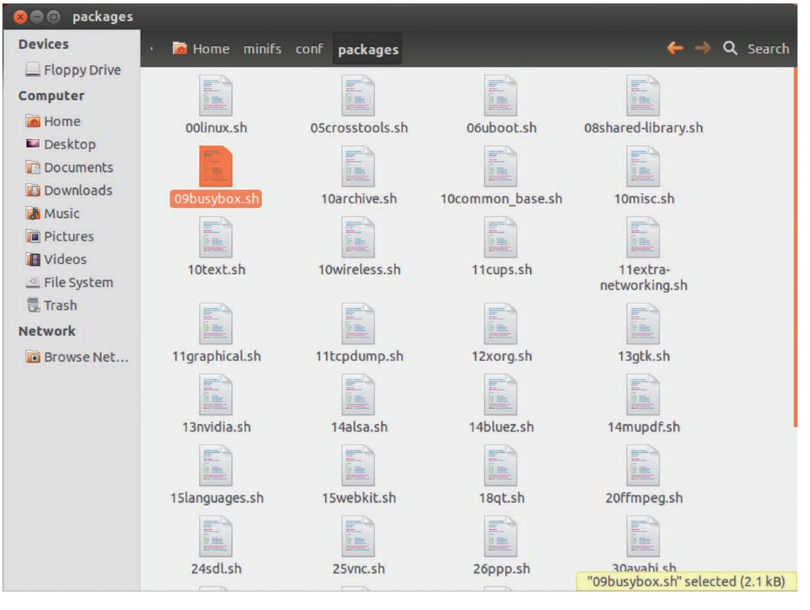
Μ' αυτή την εντολή πρόκειται να εγκατασταθούν προγράμματα, βιβλιοθήκες κι εργαλεία, που χρειάζονται για τη δημιουργία της δικής μας διανομής. Σε κάθε ερώτηση απαντάμε καταφατικά. Αφού τα εγκαταστήσουμε όλα, είμαστε έτοιμοι να προχωρήσουμε με το κατέβασμα του minifs. Θα το πάρουμε μέσω git.

```
# cd ~
# git clone https://github.com/buserror-uk/minifs.git
```

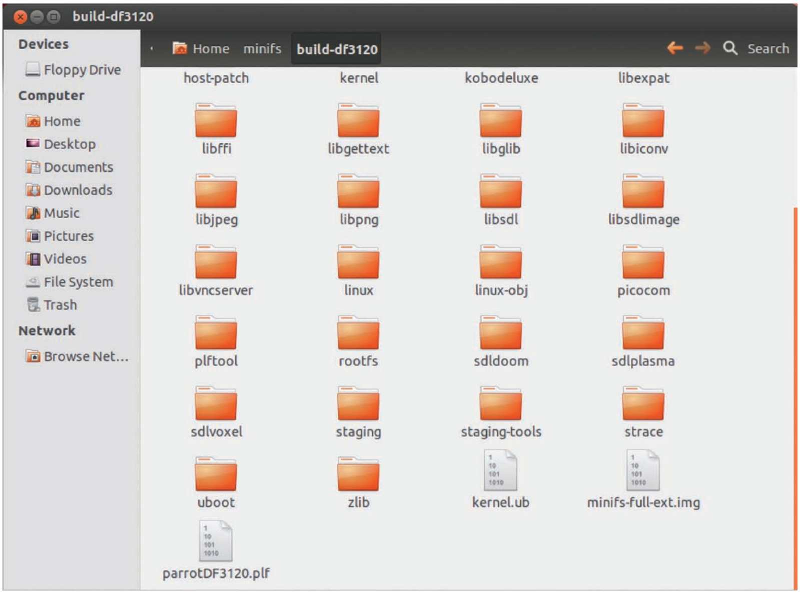
Μόλις τελειώσει το git θα έχουμε στη διάθεσή μας όλα τα απαραίτητα αρχεία και θα είμαστε έτοιμοι να δημιουργήσουμε τη δική μας διανομή Linux.

Πριν ξεκινήσουμε καλό θα ήταν να ρίξουμε μια ματιά στα αρχεία του minifs, απλά και μόνο για να πάρουμε μια ιδέα για τη λειτουργία του. Το minifs είναι ικανό να δημιουργεί «διανομές» για οποιοδήποτε υπολογιστικό σύστημα κι αρχιτεκτονική θέλει κάποιος. Μιλάμε τώρα για μικρές συσκευές έως και κανονικούς υπολογιστές, σαν αυτούς που έχουμε όλοι στο σπίτι μας. Μόνη απαίτηση του minifs είναι να υπάρχει ο κατάλληλος μεταγλωττιστής (compiler) και φυσικά η συσκευή να υποστηρίζεται από το Linux.

Όλες οι ρυθμίσεις βρίσκονται στον φάκελο conf. Δίνουμε προσοχή στους φακέλους board και packages, μια κι αυτοί θα μας χρειαστούν σε περίπτωση που θελήσουμε να πειράξουμε τα βασικά μέρη της διανομής μας. Τα BASH scripts που θα βρούμε μέσα στους φακέλους αυτούς τροποποιούν τη λειτουργία του minifs, επομένως και την μεταγλώττιση και ρύθμιση των προγραμμάτων για την κορνίζα. Καλό θα ήταν για αρχή να μην



Σε περίπτωση που θέλουμε να τροποποιήσουμε τη διαδικασία μεταγλώττισης ή να αφαιρέσουμε κάποιο πρόγραμμα από την διανομή εντελώς, ο φάκελος packages είναι αυτός που θα μας χρειαστεί.



Το minifs έχει ολοκληρώσει τη μεταγλώττιση και έχει δώσει τα τρία απαραίτητα αρχεία. Για την ακρίβεια εμείς χρειαζόμαστε μόνο τα δύο, αφού το kernel εμπεριέχεται μέσα στο σύστημα αρχείων!

πειράξουμε τίποτα απολύτως. Μπορούμε ωστόσο, ελεύθερα, να ανοίξουμε κάποιο αρχείο για να μελετήσουμε τα περιεχόμενά του.

Αφού έχουμε όλα τα απαραίτητα αρχεία διαθέσιμα και μιλήσαμε επιγραμματικά για τη λειτουργία του minifs, ας δούμε και στην πράξη τη λειτουργία του. Για αρχή θα χρειαστεί να ορίσουμε στο περιβάλλον μας την μεταβλητή MINIFS_BOARD. Η εν λόγω μεταβλητή χρησιμοποιείται από το minifs για την επιλογή της πλατφόρμας. Όπως αναφέραμε και προηγουμένως, το minifs υποστηρίζει αρκετές πλατφόρμες — μεταξύ άλλων και την κορνίζα μας. Στη συνέχεια απλά τρέχουμε το script minifs_build.sh και περιμένουμε *πολλή* ώρα, μέχρι να τελειώσει. Όχι, το εννοούμε. Στις δοκιμές μας πήρε κάτι παραπάνω από μια ώρα, για την προετοιμασία του όλου περιβάλλοντος!

```
# cd minifs
# export MINIFS_BOARD=df3120
# ./minifs_build.sh
```

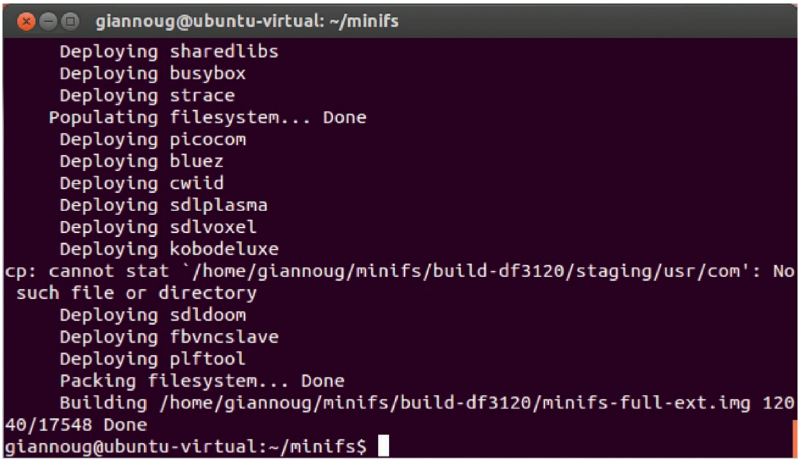
Αρχικά, θα προετοιμαστούν οι απαραίτητες βιβλιοθήκες και μεταγλωττιστές. Αυτό το στάδιο είναι που απαιτεί την περισσότερη ώρα, εδώ που τα λέμε. Προφανώς, κάθε φορά που τρέχουμε το script δε γίνονται όλα από την αρχή, αλλά μόνο τα βήματα για τα προγράμματα που πειράξαμε. Για παράδειγμα, αν θελήσουμε να μεταγλωττιστεί πάλι το busybox[5], θα πρέπει να σβήσουμε είτε τον φάκελο busybox από το build-df3120 είτε να σβήσουμε όλα τα αρχεία που ξεκινούν με τελεία και κάτω παύλα.

```
# cd ~
# rm build-df3120/busybox/._*
```

Αυτά τα αρχεία δημιουργούνται από το minifs όταν το τρέχουμε, χρησιμεύουν δε στο να γνωρίζει τι είναι έτοιμο και τι όχι.

Αφήνουμε λοιπόν το minifs να ολοκληρώσει τη δουλειά του και παίρνουμε

5. Το busybox περιέχει όλα τα καλούδια των UNIX συστημάτων, όπως το shell κι όλες τις εντολές που έχουμε συνηθίσει. Σκεφτείτε το σαν το BASH, αλλά για μικρές συσκευές.



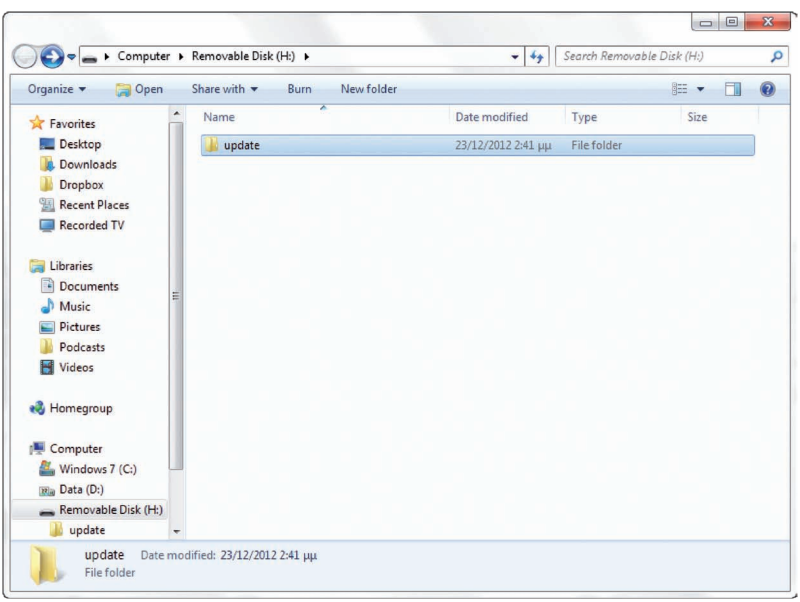
Μετά από πολλή υπομονή, το minifs ολοκλήρωσε την λειτουργία του. Κατά τη διάρκεια των δοκιμών μας χρειάστηκε μία περίπου ώρα, αλλά αυτό ίσως οφείλεται στο ότι δουλέψαμε σε virtual machine.

μια γεμάτη κι αχνιστή κούπα καφέ. Ή πορτοκαλάδα. Ή νερό. Αρκεί να αποφασίσουμε πριν τελειώσει. Μετά το τέλος της διαδικασίας θα υπάρχουν τρία αρχεία στον φάκελο build-df3120. Για τα αρχεία αυτά θα μιλήσουμε αμέσως μετά.

Εγκατάσταση uboot

Για να μπορέσουμε να ξεκινήσουμε τη διανομή μας στην ψηφιακή κορνίζα, θα πρέπει *κάπως* να δείξουμε στη συσκευή πού υπάρχουν τα αρχεία αλλά και πώς μπορούν να χρησιμοποιηθούν.

Ο δημοφιλέστερος —αν όχι ο μοναδικός— bootloader για τέτοιες συσκευές, είναι το uboot. Τον συναντάμε σε routers, κινητά τηλέφωνα αλλά και σε κάποια νέα laptops, που βασίζονται σε επεξεργαστές ARM. Είναι, με λίγα λόγια, το σημαντικότερο εξάρτημα για την εκκίνηση του πυρήνα,



Αφού συνδέσουμε την κορνίζα στον υπολογιστή, φτιάχνουμε έναν φάκελο και τοποθετούμε το αρχείο για την «αναβάθμιση» της συσκευής. Η εν λόγω αναβάθμιση περιέχει τον δικό μας bootloader, ώστε να ξεκινάμε το Linux!



Κρατώντας τον συνδυασμό πλήκτρων η συσκευή ξεκινάει τη διανομή μας! Στην πάνω αριστερή γωνία, παρατηρούμε τον Tux ;)

αφού αναλαμβάνει τη βασική αρχικοποίηση όλου του συστήματος. Ευτυχώς για εμάς, η εγκατάσταση του uboot γίνεται αρκετά εύκολα χρησιμοποιώντας τη διαδικασία αναβάθμισης της κορνίζας μας. Θα το εγκαταστήσουμε, δηλαδή, ως μια «ψεύτικη» αναβάθμιση, για το βασικό λειτουργικό της συσκευής. Αν δεν υπήρχε αυτή η δυνατότητα θα χρειαζόταν να επέμβουμε στη συσκευή με κάποιον ειδικό προγραμματιστή. (Είπε κανείς JTAG;)

Το uboot, το πρόγραμμα που μετατρέπει το uboot σε αρχείο κατάλληλο για την κορνίζα αλλά και το ίδιο το σύστημα αρχείων, έχουν δημιουργηθεί ήδη από το minifs και μπορούμε να τα βρούμε στον φάκελο build-df3120. Είναι τα αρχεία που αναφέραμε προηγουμένως.

Συνδέουμε τη συσκευή, όπως θα τη συνδέαμε για *κανονική* λειτουργία. Βάζουμε δηλαδή τον φορτιστή και γυρνάμε το διακόπτη της κορνίζας στη θέση On. Στη συνέχεια συνδέουμε τη συσκευή με καλώδιο σε μια ελεύθερη θύρα USB του υπολογιστή μας. Αμέσως, στην οθόνη θα εμφανιστεί μήνυμα πως η συσκευή είναι συνδεδεμένη σε υπολογιστή, ενώ η μνήμη θα εμφανιστεί ως εξωτερική μονάδα αποθήκευσης. Μέσα στο δίσκο αυτό δημιουργούμε έναν φάκελο με όνομα update. Εκεί, αντιγράφουμε το αρχείο parrotDF3120.plf από τον φάκελο build-df3120. Μόλις αποσυνδέσουμε το καλώδιο, η συσκευή αμέσως θα προσπαθήσει να αναβαθμίσει το λειτουργικό, εγκαθιστώντας το uboot μόνιμα στη μνήμη της! Η ενεργοποίηση του uboot γίνεται κρατώντας πατημένα το μεσαίο και το αριστερό κουμπί της συσκευής, κατά τη διάρκεια της εκκίνησης. Πριν γίνει όμως αυτό, θα πρέπει να ετοιμάσουμε την κάρτα SD.

Προετοιμασία κάρτας SD

Χρησιμοποιώντας το uboot μπορούμε πλέον να ξεκινήσουμε οποιοδήποτε Linux kernel θέλουμε μέσα από την κάρτα SD, αρκεί φυσικά να είναι συμβατό. Αφού πρόκειται για μια πολύ περιορισμένη συσκευή κι αρκετά αδύναμη, μπορούμε να χρησιμοποιήσουμε κάποια αρκετά παλιά κάρτα. Εμείς π.χ. είχαμε μια χωρητικότητας 128MB, από το μακρινό 2005!



Χρησιμοποιώντας το telnet έχουμε πρόσβαση στο κέλυφος της κορνίζας. Στον κατάλογο /usr/bin βλέπουμε όλα τα προγράμματα που έχει συμπεριλάβει ο δημιουργός του minifs.

Η κάρτα αυτή καθόλου δεν μας πειράζει πώς θα είναι χωρισμένη (π.χ., αν θα έχει ένα, δύο ή περισσότερα partitions). Μας αρκεί ένα partition στην αρχή, με μέγεθος τουλάχιστον 32MB. Αν κάποιος θέλει μπορεί ελεύθερα να δημιουργήσει ακόμη ένα partition, το οποίο θα χρησιμοποιηθεί ως swap. Ή ακόμη ένα, που θα χρησιμοποιείται για αποθήκευση δεδομένων. Ή και τα δύο. Αυτό μπορεί να γίνει αρκετά εύκολα και με ελάχιστα κλικ, χρησιμοποιώντας το GParted στο Linux σύστημά μας.

Αφού συνδέσουμε την κάρτα στο σύστημά μας – χρησιμοποιώντας κάποιο card reader – θα εμφανιστεί στο λειτουργικό ως συσκευή αποθήκευσης. Το πρώτο βήμα είναι να εντοπίσουμε ποιο γράμμα της δόθηκε από το λειτουργικό. Συνήθως θα είναι το τελευταίο αλφαβητικά.

```
# ls /dev/sd*
```

Στις δοκιμές μας, η συσκευή αυτή ήταν το sdb, με το sda να είναι ο σκληρός δίσκος του συστήματος. Φυσικά, δεν χρειάζεται ν' αναφέρουμε πως αν χρησιμοποιήσουμε λάθος γράμμα θα καταστρέψουμε τα δεδομένα της συσκευής μας! Με την κάρτα λοιπόν να είναι στο sdb, το πρώτο partition είναι το sdb1. Χρησιμοποιούμε το εργαλείο dd, ώστε να αντιγράψουμε το image των αρχείων που δημιουργήθηκε στον φάκελο build-df3120:

```
# dd if=~/minifs/build-df3120/minifs-full-ext.img of=/dev/sdb1
```

Ανάλογα με την ταχύτητα της κάρτας, η διαδικασία θα πάρει από ελάχιστα δευτερόλεπτα έως μερικά λεπτά (ειδικά σε αρκετά παλιές κάρτες). Είμαστε σχεδόν έτοιμοι. Μένει να βάλουμε την κάρτα πάλι στη συσκευή.

Εδώ μπορούμε να αποσυνδέσουμε τελείως τη συσκευή από τον υπολογιστή μας, βγάζοντας το καλώδιο USB. Έχουμε συνδεδεμένο το τροφοδοτικό και τον διακόπτη στη θέση Off. Βάζουμε την κάρτα στην συσκευή, κρατάμε πατημένα το αριστερό και το μεσαίο κουμπί και γυρνάμε τον διακόπτη στη θέση On. Για λίγα δευτερόλεπτα η οθόνη θα γεμίσει σκουπίδια[6] και, αμέσως μετά, θα εμφανιστεί το αγαπημένο μας λογότυπο του Linux: ο Tux!

Παιχνίδια στην κορνίζα

Η κορνίζα μας είναι ρυθμισμένη ώστε να χρησιμοποιεί το usb-ethernet-gadget, μια λειτουργία του Linux kernel που επιτρέπει επικοινωνία δικτύου Ethernet μέσα από θύρα USB. Έτσι, αν τώρα συνδέσουμε τη συσκευή στον υπολογιστή μας, θα δούμε ότι θα εμφανιστεί ως μια συσκευή δικτύου. Το μόνο που μένει είναι να δώσουμε «με το χέρι» μια διεύθυνση IP στο κατάλληλο interface του λειτουργικού και ύστερα να συνδεθούμε, μέσω telnet, στην κορνίζα. Συνδέουμε λοιπόν την κορνίζα και ξεκινάμε:

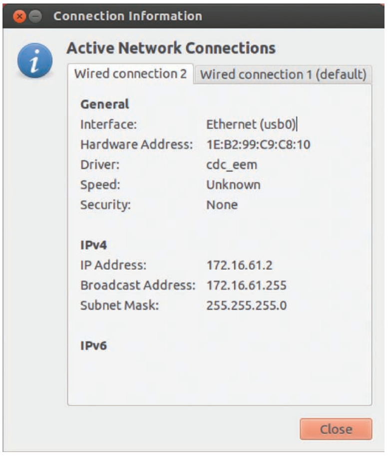
```
# ifconfig usb0 172.16.61.2 netmask 255.255.255.0 up
# telnet 172.16.61.1
```

Είμαστε πλέον *μέσα* στην κορνίζα! Αν αναρωτιέστε γιατί συνδεθήκαμε με το γεμάτο τρύπες και καθόλου ασφαλές telnet, σας υπενθυμίζουμε ότι η συσκευή έχει μόλις 8MB RAM.

Υπάρχουν μερικά ακόμη προγραμματάκια που μπορούμε να δοκιμάσουμε, πριν ασχοληθούμε περισσότερο. Το πιο εντυπωσιακό πιστεύουμε πως είναι το newvox, το οποίο μπορούμε να απολαύσουμε απλά δίνοντας newvox στο τερματικό. Δυστυχώς δεν μπορούμε να το απολαύσουμε πλή-

6. Με τον όρο «σκουπίδια» οι προγραμματιστές αναφέρονται σε τιμές της μνήμης που δεν γνωρίζουμε τι αντιπροσωπεύουν. Για παράδειγμα, μόλις ανοίξουμε τον υπολογιστή μας η μνήμη RAM είναι γεμάτη απ' αυτά :)

ρως αφού δεν παρέχεται κάποιος τρόπος χειρισμού[7], μπορούμε όμως να το δούμε στο YouTube. Ακόμη ένα πρόγραμμα με ωραία γραφικά είναι το plasma. Εντυπωσιακά, αν σκεφτούμε ότι τρέχουν σε μια κορνίζα.



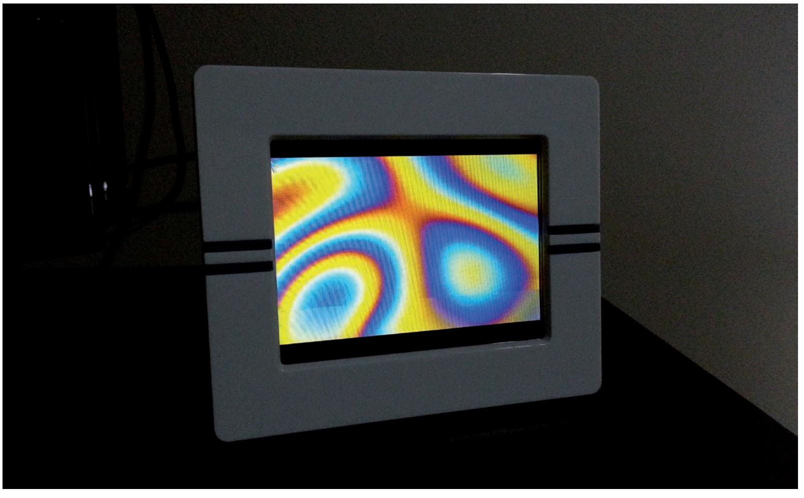
Η σύνδεση με τη συσκευή γίνεται χρησιμοποιώντας το ethernet-gadget, μια λειτουργία του Linux kernel που χρησιμοποιεί τη θύρα USB για επικοινωνία Ethernet!

Και τώρα;

Μπορεί η συσκευή μας να είναι αρκετά πιο αδύναμη σε σύγκριση με ό,τι έχουμε δουλέψει και συνηθίσει, μολαταύτα είναι ικανή για πολλά. Σημειώνουμε ότι υπάρχει υποστήριξη για τη βιβλιοθήκη SDL (το είδατε στα μηνύματα κατά τη μεταγλώττιση, έτσι δεν είναι;) καθώς κι ότι η συσκευή είναι ικανή να τρέξει μια εκδοχή του Doom, το SDLDoom!

Με αρκετή όρεξη και πολύ κώδικα, η συσκευή είναι δυνατόν να μετατραπεί σε ένα μικρό weather station. Οι δυνατότητες είναι αρκετές. Θα μπορούσαμε, π.χ., να δημιουργούμε τις εικόνες σε κάποιον απομακρυσμένο server και να τις τραβάμε, περιοδικά, μέσα από την κορνίζα. Θα μπορούσαμε βέβαια να γράψουμε ένα πρόγραμμα που δεν θα βασίζεται σε κάποιον εξωτερικό server αλλά μόνο του θα έχει τη δυνατότητα να κατεβάζει δεδομένα και να δημιουργεί γραφικά. Αυτό ακριβώς αποτελεί απώτερο σκοπό του κυρίου ο οποίος αυτή την στιγμή πατάει μανιωδώς πλήκτρα. Μόλις αποφασίσει για τις προδιαγραφές —κι ενδέχεται να έχει ήδη αποφασίσει—, θα προχωρήσει στην υλοποίηση και φυσικά στην παρουσίαση, μέσα από αντίστοιχο άρθρο.

Κι αυτό το τελευταίο, δεν ήταν παρά μια ξεκάθαρη προειδοποίηση.



Το plasma σε λειτουργία! Πρόκειται για ένα από τα δοκιμαστικά προγράμματα που περιέχει το minifs. Η συσκευή καταφέρνει γύρω στα 50fps, ταχύτητα αρκετά εντυπωσιακή για τα δεδομένα της.

7. Για την ακρίβεια υπάρχει, μια κι ένα από τα προγράμματα μπορεί να συνδεθεί με το WiiMote του Nintendo Wii και μπορούμε να χρησιμοποιήσουμε το εν λόγω χειριστήριο με την κορνίζα!

Skill: Intermediate

Tags: Observium, web app, SNMP, monitoring

ΑΠΟΛΥΤΗ ΕΠΟΠΤΕΙΑ ΤΩΝ ΜΗΧΑΝΩΝ



Είμαστε ανήσυχα πνεύματα — κι αυτό δεν είναι κρυφό. Καταπιανόμαστε με το κάθε τι — και συνήθως μπλέκουμε πολλά μαζί. Μια από τις ασχολίες μας είναι οι servers, κάθε είδους! Τα αντίστοιχα μηχανήματα/VMs ενδέχεται να βρίσκονται κοντά μας ή να είναι κάπου απομακρυσμένα, συνήθως ξεχασμένα να κάνουν κάποια συγκεκριμένη εργασία. Η εργασία αυτή μπορεί να 'ναι σημαντική, μπορεί και όχι. Σε κάθε περίπτωση, ένα από τα προβλήματα που συναντά ο ανήσυχος, ψαγμένος χρήστης, εντοπίζεται στη συντήρηση και στην παρακολούθηση όλων αυτών των servers. Δεν είναι και τόσο εύκολο να εποπτεύεις τη λειτουργία αυτών των συστημάτων, ειδικά αν πρόκειται για απομακρυσμένους servers που κάνουν αρκετά «φιλοσοφημένες» εργασίες...

by GiannouG

Αυτή τη στιγμή είναι πολύ πιθανό να σκέφτεστε το Nagios, το Cacti ή κάποιο άλλο δημοφιλές πρόγραμμα παρακολούθησης και διαχείρισης servers. Υπάρχει ένα μικρό προβληματάκι με αυτά τα προγράμματα. Μπορεί να είναι γνωστά και να χρησιμοποιούνται παντού κι από όλους τους σοβαρούς διαχειριστές, ταυτόχρονα όμως είναι κι αρκετά δύσκολα τόσο στην εγκατάσταση, όσο και στη ρύθμιση αλλά και συντήρηση. Εργαλεία σαν τα προαναφερθέντα έχουν σίγουρα νόημα σε μεγάλες εγκαταστάσεις. Όμως ο ανήσυχος, ψαγμένος χρήστης, αφενός δεν έχει στην κατοχή του τόσα μηχανήματα, αφετέρου δεν διαθέτει την πολυτέλεια χρόνου για ν' ασχοληθεί πολύ μαζί τους! Μην ξεχνάτε άλλωστε ότι είναι κι άλλα πρότζεκτ που τον περιμένουν... :)



Κάπως έτσι ένιωθε ο γράφων, όταν εγκατέστησε για πρώτη φορά το Observium. Για όσους δε γνωρίζουν, ο κύριος στα δεξιά είναι ο αρχιτέκτονας του Matrix :D

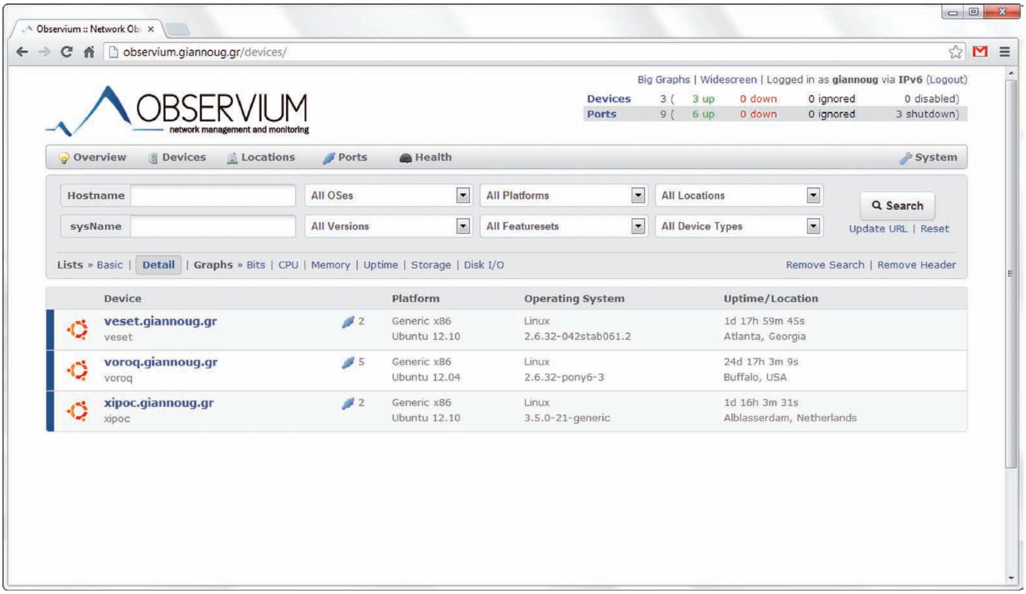
Γνωρίστε το Observium!



Η μασκότ του Observium είναι ένα τρισχαριτωμένο χαμστεράκι, με γυαλάκια που το βοηθούν να βλέπει τα πάντα :D Όχι τα αρκουδάκια «πάντα». (Σ.τ.Ε. Έπεσα στα πατώματα!)

Το Observium σε καμία περίπτωση δεν φιλοδοξεί ν' αντικαταστήσει τα επαγγελματικά εργαλεία παρακολούθησης και ειδοποίησης. Μολαταύτα, μπορεί να παρουσιάσει γραφήματα και την κατάσταση των υπό εποπτεία μηχανημάτων πολύ καλύτερα σε σχέση με άλλες λύσεις — και μάλιστα με ωραιότερο τρόπο. Σκοπός του Observium είναι να προσφέρει όσο το δυνατόν περισσότερες πληροφορίες, με τη μικρότερη δυνατή παρέμβαση από πλευράς χρήστη. Επιπλέον, χρησιμοποιώντας το πρωτόκολλο SNMP είναι σε θέση να αναγνωρίζει αυτόματα το hardware των συστημάτων και να παρέχει, εντελώς αυτοματοποιημένα, όλα τα γραφήματα που έχουν νόημα. Το μόνο που χρειαζόμαστε για να παρακολουθήσουμε ένα μηχάνημα, είναι ένας SNMP server σ' αυτό!

Το Observium είναι μια εφαρμογή γραμμένη σε PHP και για τη λειτουργία της απαιτεί κάποιον web server με τον απαραίτητο διερμηνευτή (interpreter) της PHP εγκατεστημένο στο σύστημα. Για τις ανάγκες του άρθρου καταφύγαμε στον Apache, μια και είναι ο ευκολότερος στην ρύθμιση. Να σημειώσουμε επίσης ότι το Observium το τρέχουμε εδώ και



Η προσωπική εγκατάσταση του γράφοντα. Κρατούνται γραφήματα και ιστορικά για τρία «σοβαρά» μηχανήματα, ενώ υπάρχουν alarms σε περίπτωση που κάποιο πέσει!

μήνες σε nginx — και λειτουργεί απροβλημάτιστα. Με την ευκαιρία, να υπογραμμίσουμε εδώ ότι το Observium δεν θέλει να βρίσκεται σε υποκαταλόγους κάποιου domain: Αυστηρά, πρέπει να βρίσκεται στο root. Αυτό σημαίνει ότι θα πρέπει να το κάνουμε να φαίνεται είτε στη διεύθυνση IP ενός μηχανήματος είτε σε κάποιο domain. Για παράδειγμα, αν προσπαθήσουμε να το τρέξουμε από το example.com/observium, θα δούμε πως, εχμ, τίποτα! Εντάξει, τίποτα εκτός από κάτι μαύρα γράμματα[1] :S

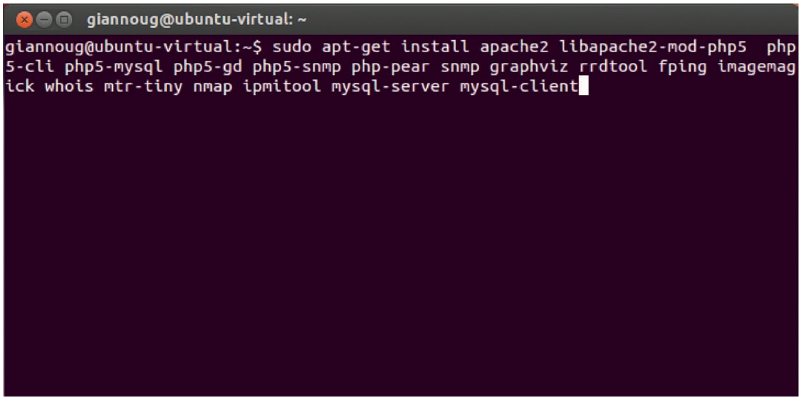
Εγκατάσταση web server και βασικού συστήματος

Όπως έχει γίνει φανερό, το Observium δεν είναι τίποτε άλλο από μερικά PHP scripts. Χρειαζόμαστε λοιπόν την PHP, έναν web server αλλά και μια βάση δεδομένων, η οποία θα χρησιμοποιείται για αποθήκευση δεδομένων από το Observium. Αλήθεια, υπάρχει σοβαρό PHP script που δεν χρησιμοποιεί τη MySQL ή κάποια άλλη βάση; Δεν υπάρχει! Αν έχουμε κάποιον έτοιμο webserver, δεν είναι αναγκαστικό να βάλουμε τον Apache. Το μόνο που θα χρειαστεί να τον ρυθμίσουμε κατάλληλα. Ξεκινάμε με την εγκατάσταση των πακέτων, σε σύστημα Ubuntu 12.10 και φυσικά από την κονσόλα:

```
$ sudo apt-get install apache2 libapache2-mod-php5  php5-cli
php5-mysql php5-gd php5-snmp php-pear snmp graphviz rrdtool
fping imagemagick whois mtr-tiny nmap ipmitool mysql-server
mysql-client
```

Αυτά είναι όλα τα βασικά πακέτα που θα χρειαστούμε για τη φιλοξενία και λειτουργία του Observium. Σε κάθε ερώτηση αφήνουμε τις προεπιλογές και, όπου μας ζητηθεί κωδικός, δίνουμε κάτι αρκετά πολύπλοκο που φυσικά το θυμόμαστε. (Προσέξτε μόνο να μην την πάθετε όπως ο τύπος στο deltaCast s01e04: http://deltahacker.gr/?p=7485.)

1. Αυτό γιατί οι προγραμματιστές έχουν χρησιμοποιήσει περίεργους τρόπους για τον εντοπισμό των απαραίτητων CSS και JavaScript files. Βασικά το περίεργο είναι πως δεν θέλουν να το διορθώσουν! Σίγουρα κάτι παραπάνω θα ξέρουν, βέβαια :D



Η εγκατάσταση ξεκινά με τα βασικά πακέτα που χρειάζονται για τη λειτουργία του Observium. Μεταξύ αυτών είναι κι εκείνο του Apache, του web server που επιλέξαμε για τη συγκεκριμένη παρουσίαση.

Επιπλέον, θα χρειαστούμε μερικές βιβλιοθήκες της PHP. Η εγκατάσταση των βιβλιοθηκών γίνεται μέσα από το pear, ένα πρόγραμμα διαχείρισης και συγκέντρωσης βιβλιοθηκών για τη γλώσσα PHP. Επίσης, «pear» λέγεται και η συλλογή των βιβλιοθηκών αυτών!

```
$ sudo pear install Net_IPv6
$ sudo pear install Net_IPv4
```

Αφού φέραμε το σύστημα στα μέτρα του Observium, ώρα να ρυθμίσουμε και το τελευταίο κομμάτι της πλατφόρμας: τη βάση δεδομένων, στην οποία θα αποθηκεύονται όλα τα στοιχεία που συλλέγει η εφαρμογή. Για την ώρα δεν θα δημιουργήσουμε κάποιον διαφορετικό χρήστη. Το Observium θα συνδέεται μέσω του χρήστη root της MySQL.

```
$ mysql -u root -p
```

(δίνουμε τον κωδικό του χρήστη root, που ορίσαμε στην εγκατάσταση)

```
mysql> CREATE DATABASE observium;
```

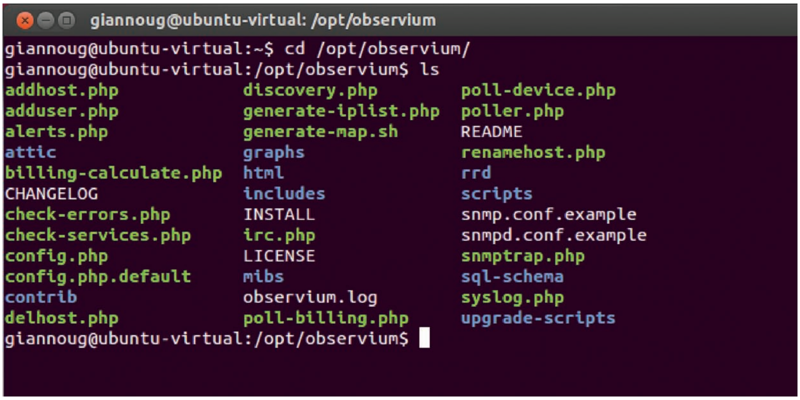
Η βάση δημιουργήθηκε, αν και προς το παρόν είναι άδεια. Τελειώσαμε με την προετοιμασία του συστήματος. Ο Apache είναι έτοιμος, η PHP με τις απαραίτητες βιβλιοθήκες εγκαταστάθηκε, δημιουργήσαμε και τη βάση για την αποθήκευση δεδομένων. Σειρά έχει το Observium και οι τελευταίες ρυθμίσεις για τη σωστή λειτουργία του.

Έρχεται!

Το σύστημά μας είναι έτοιμο να υποδεχτεί το Observium. Την εγκατάστασή του θα την κάνουμε από το SVN, μια και η ομάδα ανάπτυξης έχει επιλέξει αυτό τον τρόπο για τη διανομή του software. Το SVN χρησιμοποιείται κυρίως από τους προγραμματιστές μιας ομάδας, για τη διαχείριση των εκδόσεων της εφαρμογής πάνω στην οποία δουλεύουν. Οι περισσότερες, αν όχι όλες, οι διανομές Linux, δεν έχουν εγκατεστημένα τα απαραίτητα εργαλεία για την επικοινωνία με SVN repositories. Το πακέτο που περιέχει όλα τα απαραίτητα προγράμματα και βιβλιοθήκες στα Debian-based OSes ονομάζεται subversion και μπορούμε να το εγκαταστήσουμε γρήγορα, με το apt-get:

```
$ sudo apt-get install subversion
```

Έτοιμοι! Η εγκατάσταση του Observium θα γίνει στον φάκελο /opt/observium. Ο φάκελος /opt χρησιμοποιείται συνήθως για προγράμματα που εγκαθίστανται εξωτερικά, δηλαδή χωρίς τη χρήση του επίσημου



Η εγκατάσταση του Observium έγινε μέσω SVN και τα σχετικά αρχεία αποθηκεύτηκαν στο /opt/observium. Ό,τι πηγαίνει στο /opt συνήθως δεν έρχεται μέσω του επίσημου package manager.

package manager της εκάστοτε διανομής. Στον κατάλογο αυτό εγκαθίστανται και προγράμματα που δεν είναι ανοιχτού κώδικα, όπως, π.χ., το Oracle JVM.

```
$ cd /opt
$ sudo svn checkout \ http://www.observium.org/svn/observer/
trunk/ observium
```

Τα απαραίτητα αρχεία θα κατέβουν, όλα στην τελευταία τους έκδοση. Συνεχίζουμε με την ρύθμιση του Observium και των προγραμμάτων που εγκαταστήσαμε πριν. Θα δουλέψουμε αυστηρά από το τερματικό κι όχι από τον web browser μας, όπως ίσως έχουμε συνηθίσει με άλλες PHP-based εφαρμογές (το WordPress είναι το πρώτο παράδειγμα που μας έρχεται στο μυαλό).

```
$ cd observium
$ sudo cp config.php.default config.php
```

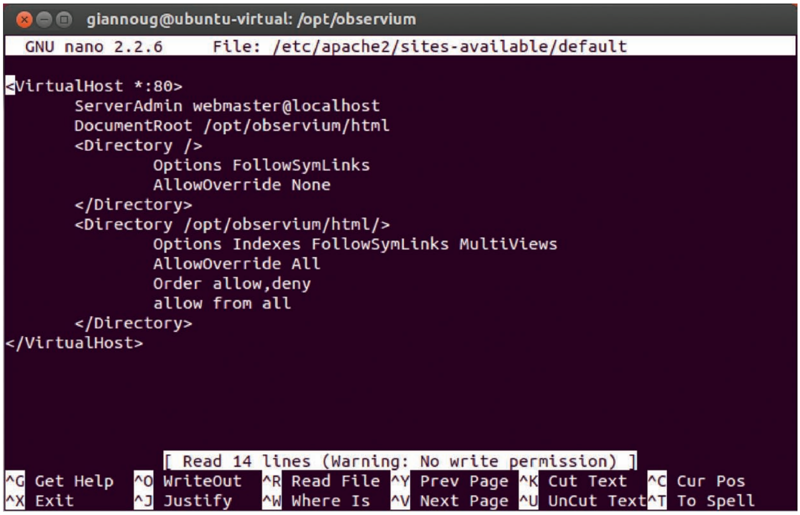
Ανοίγουμε το config.php με κάποιον text editor, όπως είναι το nano, και η ρύθμιση ξεκινά! Για αρχή βεβαιωνόμαστε ότι οι παράμετροι για τη σύνδεση με τη MySQL είναι σωστές. Πιο συγκεκριμένα, θα χρειαστεί να αλλάξουμε το όνομα χρήστη, τον κωδικό πρόσβασης αλλά και το όνομα της βάσης με αυτά που δώσαμε προηγουμένως. Το όνομα χρήστη είναι root, ο κωδικός είναι αυτός που δώσαμε κατά την εγκατάσταση της βάσης και, τέλος, το όνομά της παραμένει ως έχει. Δείτε τώρα την ακόλουθη γραμμή:

```
php includes/sql-schema/update.php
```

Η ρύθμιση των πινάκων της βάσης γίνεται αυτόματα, μέσα από το script update.php. Τυχόν errors δεν μας ανησυχούν κι απλά τα αγνοούμε. Εγκαταλείπουμε τον text editor και συνεχίζουμε με τη δημιουργία κάποιων βασικών φακέλων.

```
$ sudo mkdir graphs rrd
$ sudo chown -R www-data:www-data /opt/observium
```

Οι φάκελοι graphs και rrd είναι απαραίτητοι για τα γραφήματά μας. Χωρίς αυτούς, το Observium δε θα λειτουργεί. Για την ακρίβεια, απλά δε θα δείχνει τίποτα — ούτε καν κάποιο σφάλμα. Σιγουρευόμαστε λοιπόν ότι οι φάκελοι υπάρχουν, καθώς κι ότι ο χρήστης www-data του συστήματος είναι ο ιδιοκτήτης τόσο των φακέλων αυτών, όσο και όλων των αρχείων του Observium.



Ρυθμίσαμε τον Apache ώστε να σερβίρει μονάχα το Observium.

Μεταφερόμαστε τώρα στο αρχείο ρυθμίσεων του Apache. O server αυτός εγκαταστάθηκε αποκλειστικά για να σερβίρει το Observium. Θέλουμε, δηλαδή, να σερβίρει τις σχετικές σελίδες και τίποτε άλλο, για αυτό και θα χαλάσουμε τις προεπιλεγμένες του ρυθμίσεις. Ανοίγουμε το default αρχείο ρυθμίσεων:

```
$ sudo nano /etc/apache2/sites-available/default
```

Σβήνουμε όλα τα περιεχόμενα και προσθέτουμε τα ακόλουθα:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /opt/observium/html
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /opt/observium/html/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>
</VirtualHost>
```

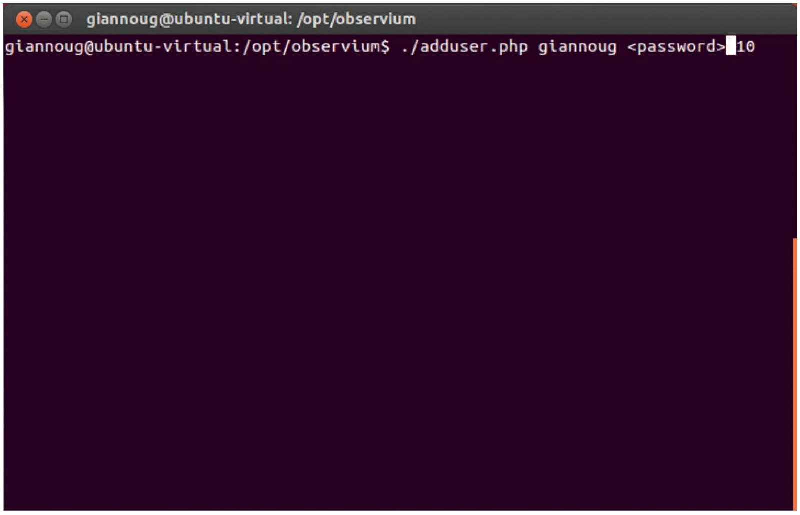
Αποθηκεύουμε τις αλλαγές, εγκαταλείπουμε τον editor και συνεχίζουμε με την ενεργοποίηση του mod_rewrite. Το εν λόγω module βοηθάει τον Apache και το Observium, με την παραμετροποίηση των URL. Ολοκληρώνουμε, φυσικά, με μια επανεκκίνηση του web server.

```
$ sudo a2enmod rewrite
```

```
$ sudo service apache2 restart
```

Λίγο πριν τελειώσουμε μια και καλή με την εγκατάσταση του Observium, θα χρειαστεί να κάνουμε δύο ακόμη δουλειές. Πρώτα απ' όλα χρειάζεται να ορίσουμε έναν νέο χρήστη για το Observium, σε ρόλο διαχειριστή:

```
$ ./adduser.php <username> <password> 10
```



Για την προσθήκη κάποιου χρήστη στο Observium, θα τρέξουμε το script adduser.php. Για την προσθήκη διαχειριστή, θα πρέπει επιπλέον να δώσουμε, ως επίπεδο δικαιωμάτων, το 10!

Ακολούθως θα πρέπει να ορίσουμε κάποια cron jobs, έτσι ώστε τα δεδομένα να μαζεύονται αυτόματα από το Observium χωρίς τη δική μας παρέμβαση. Ανοίγουμε το αρχείο /etc/cron.d/observium:

```
$ sudo nano /etc/cron.d/observium
```

Μέσα, θα χρειαστεί να προσθέσουμε τις παρακάτω γραμμές. Χρειάζονται τρία διαφορετικά scripts για **α)** τον εντοπισμό νέων δικτυακών συσκευών, **β)** τον εντοπισμό νέων χαρακτηριστικών των συσκευών μας και **γ)** τη συλλογή των δεδομένων.

```
33 */6 * * * root /opt/observium/discovery.php -h all >> /dev/
null 2>&1
*/5 * * * * root /opt/observium/discovery.php -h new >> /dev/
null 2>&1
*/5 * * * * root /opt/observium/poller.php -h all >> /dev/null
2>&1
```

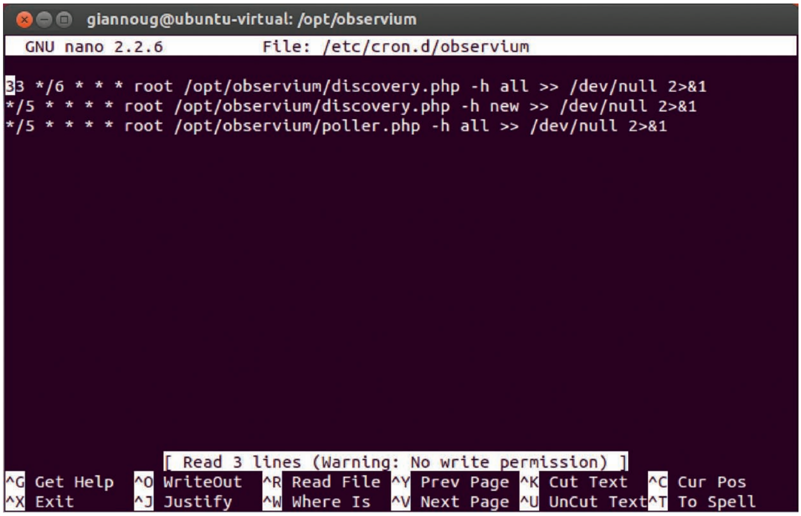
Τελειώσαμε μια και καλή με την εγκατάσταση του Observium. Μπορούμε, τώρα, να ανοίξουμε έναν web browser και να τον στείλουμε στη διεύθυνση IP του μηχανήματος που τρέχει το Observium. Αυτό θα μας υποδεχθεί ζητώντας να συνδεθούμε. Το κάνουμε και ...αντιλαμβανόμαστε πως δεν υπάρχει τίποτα ακόμη. Λογικό, αφού δεν υπάρχουν συνδεδεμένοι clients! Προχωράμε;

Η εγκατάσταση των clients

Περνάμε στην αντίπερα όχθη, τώρα, εκεί που βρίσκονται οι clients. Ο server, δηλαδή το ίδιο το Observium, είναι έτοιμος. Αυτό που χρειαζόμαστε είναι οι clients, από τους οποίους θα μπορούμε να αντλούμε δεδομένα. Ένας client μπορεί να είναι είτε το ίδιο το μηχάνημα που σερβίρει το Observium είτε κάποιος απομακρυσμένος server είτε οποιαδήποτε άλλη συσκευή ξέρει το πρωτόκολλο SNMP. Για την ώρα, θα δούμε πώς μπορούμε να ρυθμίσουμε κατάλληλα κάποιο άλλο μηχάνημα ή εκείνο που σερβίρει το Observium.

```
$ sudo apt-get install snmpd
```

Όλη η «βρώμικη» δουλειά γίνεται από τις τρεις αυτές γραμμές... Τα συγκεκριμένα cron jobs τρέχουν ανά τακτά χρονικά διαστήματα κι ενημερώνουν τόσο τις διαθέσιμες συσκευές, όσο και τα δεδομένα που λαμβάνονται από αυτές.



Η εγκατάσταση του SNMP daemon γίνεται όπως συνηθίζεται στο λειτουργικό πάνω στο οποίο δουλεύουμε — και το παράδειγμα που μόλις δώσαμε ισχύει σε Debian-based διανομές. Θα χρειαστούν μερικά ακόμη βήματα για να κάνουμε τον daemon να μιλά σωστά με το Observium. Ανοίγουμε το αρχείο /etc/default/snmpd κι εντοπίζουμε τον ορισμό του SNMPDOPTS. Για να αποφύγουμε κάποια ανούσια μηνύματα του στυλ «συνδέθηκε πελάτης την τάδε ώρα», φροντίζουμε ώστε να ισχύει το ακόλουθο:

```
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -p /var/run/snmpd.pid'
```

Σειρά έχει η βασική ρύθμιση του snmpd, εκείνη που χαρακτηρίζει τη λειτουργία του και τα δεδομένα που θα μεταφέρονται με κάθε αίτηση από πλευράς Observium. Θα χρειαστεί να αλλάξουμε τις ρυθμίσεις σε τρια διαφορετικά σημεία, τα οποία σχολιάζουμε ακολούθως.

```
com2sec readonly default <COMMUNITY>
group MyROGroup v1 readonly
group MyROGroup v2c readonly
group MyROGroup usm readonly
view all included .1 80
access MyROGroup "" any noauth exact all none none
syslocation <LOCATION>
syscontact <CONTACT>
extend .1.3.6.1.4.1.2021.7890.1 distro /usr/bin/distro
```

Δίνουμε προσοχή στα <COMMUNITY>, <LOCATION> και <CONTACT>. Κατ' αρχάς, ξεκινάμε με το community. Το SNMP λειτουργεί με τα λεγόμενα communities κι ένα συνηθισμένο είναι το public, που επιτρέπει σε όλους να δουν τα δεδομένα που μεταφέρονται μέσω SNMP. Εμείς θα χρειαστεί να ορίσουμε κάτι άλλο — οτιδήποτε θέλουμε. Καλό θα ήταν να δώσουμε κάτι όχι τόσο συνηθισμένο, ειδικά αν δε θέλουμε να βλέπει κανείς άλλος τις πληροφορίες του συστήματός μας.

Στη συνέχεια θα χρειαστεί να αλλάξουμε τα <LOCATION> και <CONTACT>. Το πρώτο χρησιμοποιείται από το Observium για να χωρίσει τα μηχανήματα ανάλογα την περιοχή που βρίσκονται, ενώ το δεύτερο χρησιμοποιείται όταν χρειάζεται να επικοινωνήσει με κάποιον. Για παράδειγμα,

στο <LOCATION> θα μπορούσαμε να βάλουμε «Athens, Greece» για τον server που έχουμε κάτω από το γραφείο, ενώ στο <CONTACT> κάποιο email.

Τέλος εγκαθιστούμε το distro, ένα μικρό προγραμματάκι που βοηθά το Observium να εντοπίζει τη διανομή του συστήματος εύκολα, γρήγορα και πάνω από όλα σωστά!

```
$ sudo wget http://www.observium.org/svn/observer/trunk/scripts/
distro
$ sudo mv distro /usr/bin/distro
$ sudo chmod 755 /usr/bin/distro
```

Επανεκκινούμε το SNMP service και σταυρώνουμε τα δάχτυλα...

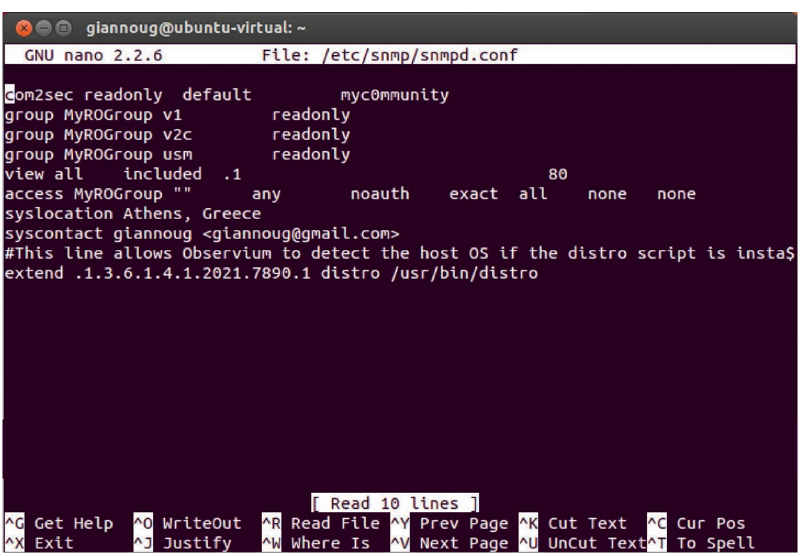
```
$ sudo service snmpd restart
```

Να πούμε εδώ ότι τα παραπάνω βήματα μπορούμε να τα επαναλάβουμε σε όλα τα μηχανήματα που θέλουμε να παρακολουθούμε: τυπικούς servers, desktop συστήματα, VPSes, virtual machines — ακόμη και στο μικρό μας Raspberry Pi!

Ωραία γραφηματάκια

Μπορούμε πλέον να προσθέσουμε τη συσκευή ή τις συσκευές μας στο Observium. Ανοίγουμε το *άδειο* web panel, συνδεόμαστε και πηγαίνουμε στο Device → Add Device. Εκεί θα μας ζητηθεί τόσο το hostname όσο και το SNMP community της συσκευής. Δεν χρειάζεται τίποτε περισσότερο!

Μετά από μερικά δευτερόλεπτα —κι αφού το Observium προσπαθήσει να εντοπίσει την συσκευή—, αυτή θα προστεθεί στη λίστα των συσκευών. Για αρχή, θα προσέξουμε πως στην συσκευή δεν θα εμφανίζεται καμία επιπλέον πληροφορία. Θα πρέπει να περιμένουμε έως ότου να τρέξουν τα cron jobs που ρυθμίσαμε πριν, ώστε να γίνει η συλλογή των δεδομένων. Ώρα για ένα ποτήρι νερό! (Σ.τ.Ε. Λέτε να είναι και της υγιεινής διατροφής;)



Το αρχείο ρυθμίσεων του SNMP client ορίζει το ποια δεδομένα θα μεταφέρονται στο Observium, όταν εκείνο αρχίσει τις ερωτήσεις...

Αφού μαζευτούν τα πρώτα δεδομένα, θα προσέξουμε ότι σιγά-σιγά όλα τα μενού θα γεμίζουν με πληροφορίες. Θα αφήσουμε την εξερεύνηση εξ ολοκλήρου σε εσάς, απλά θα αναφερθούμε στα βασικά μενού. Στο Overview, υπάρχουν όλες οι βασικές πληροφορίες που μας ενδιαφέρουν — ή που, τέλος πάντων, το Observium πιστεύει ότι μας ενδιαφέρουν. Εδώ παρουσιάζονται πληροφορίες π.χ. για τις όποιες επανεκκινήσεις γίνονται[2]. Στο μενού Locations, όπως προδίδει και το όνομά του, μπορούμε να δούμε τις τοποθεσίες των συσκευών μας. Στο μενού Ports εμφανίζονται όλες οι συσκευές δικτύου, μαζί με πληροφορίες όπως bandwidth και κατάσταση. Κλείνουμε με το μενού Health, όπου εκεί βλέπουμε συγκεντρωτικά την κατάσταση των επεξεργαστών, των μνημών αλλά και των συσκευών δικτύου! Κάθε μενού έχει αρκετά υπο-μενού, τα οποία περιμένουν να εξερευνηθούν. Όλα τους.

Observium update

Όλα λειτουργούν ρολόι, τα γραφήματά μας είναι πολλά κι ενδιαφέροντα και το σύστημα τρέχει χωρίς πρόβλημα. Τι γίνεται σε περίπτωση που η ομάδα του Observium κυκλοφορήσει κάποια νέα έκδοση, με πιθανώς νέα χαρακτηριστικά ή έστω διορθώσεις προβλημάτων; Το μόνο που έχουμε να κάνουμε είναι απλώς να «τραβήξουμε» τις νέες εκδόσεις των αρχείων, μέσω SVN:

```
$ cd /opt/observium
$ sudo svn update
```

Για να σιγουρευτούμε ότι το πρόγραμμα λειτουργεί σωστά με τις αλλαγές καθώς κι ότι η βάση δεδομένων είναι αναβαθμισμένη, τρέχουμε το discovery.php. Αυτό θα ελέγξει αν όλα λειτουργούν σωστά ή όχι.

```
$ ./discovery.php -h new
```

Αυτό ήταν! Έχουμε την τελευταία έκδοση του Observium, εύκολα και γρήγορα.

Συμπερασματικά...

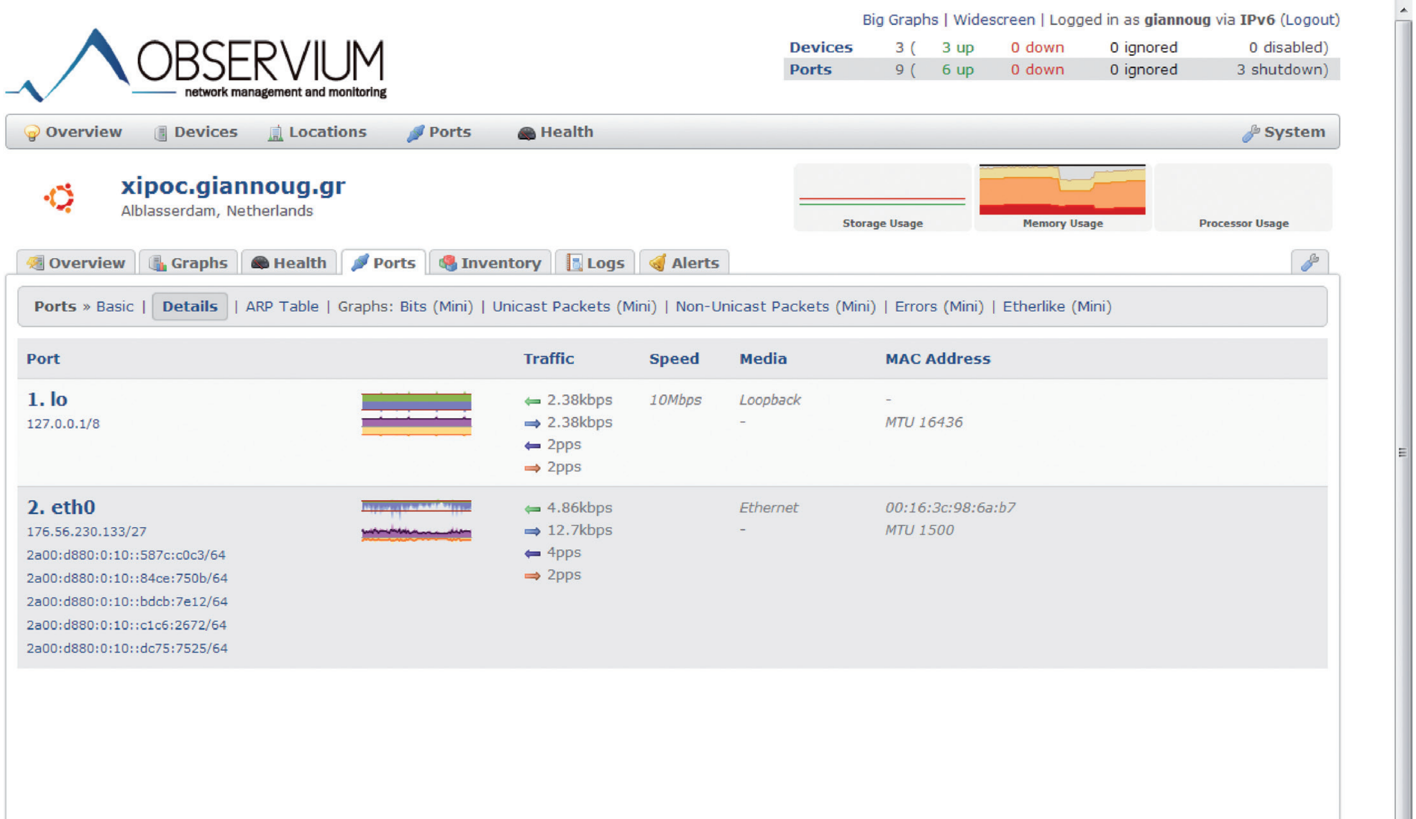
Το Observium, αν χρησιμοποιηθεί σωστά, είναι σε θέση να λύσει εντελώς τα χέρια κάθε system administrator. Ακόμη κι αν δεν διαχειριζόμαστε πολύπλοκα και κρίσιμα μηχανήματα, το Observium είναι ό,τι πρέπει για να έχουμε άριστη εποπτεία.

Στο παρόν κείμενο είδαμε μόνο τη βασική λειτουργία του Observium. Μπορούμε να εμπλουτίσουμε τα διαγράμματα αλλά και τη λειτουργία του με αρκετά ακόμη στοιχεία. Αναφέρουμε, για παράδειγμα, ότι το Observium είναι σε θέση να παρακολουθεί συγκεκριμένες εφαρμογές, όπως τον nginx και τον Apache. Τι λέτε; Θα το υποχρεώσετε να το κάνει;

2. Εν γνώσει μας ...ή όχι!



Μια από τις σημαντικότερες καρτέλες του Observium είναι η «Overview» — και υπάρχει για κάθε μηχάνημα που παρακολουθείται.



Ακόμη μία αρκετά ενδιαφέρουσα καρτέλα: Εδώ βλέπουμε τα στατιστικά κάθε συσκευής δικτύου, με ωραιότατα γραφήματα αλλά και πληροφορίες για την ταχύτητα μεταφοράς δεδομένων.

Skill: Intermediate

Tags: BASH, scripting

ΠΡΟΓΡΑΜΜΑΤΙΣΤΕ ΤΟ BASH



Κυριαρχήστε_στο_command_line:/

part 4/4

Οι δυνατότητες που προσφέρει το κέλυφος είναι (σχεδόν) απειραριθμήσιμες και παρουσιάζουν όλες ιδιαίτερο ενδιαφέρον. Ωστόσο υπάρχει μία που ξεχωρίζει — και μάλιστα με διαφορά: Πρόκειται για εκείνη που επιτρέπει το συνδυασμό και τον αυτοματισμό όλων των άλλων δυνατοτήτων!

by Spir@l Evolut10n

Η ταυτόχρονη εκτέλεση πολλών εργασιών, η ανακατεύθυνση, η διασωλήνωση, ο χρονοπρογραμματισμός με το crontab και τα aliases, αποτελούν μερικά μόνο από τα πανίσχυρα εργαλεία που προσφέρει το κέλυφος. Μεταξύ αυτών, εκείνα που θα φανούν περισσότερο χρήσιμα (και καμιά φορά εντυπωσιακά) είναι η διασωλήνωση και τα συνώνυμα. Βλέπετε, η αποστολή αυτών των δύο είναι αφηρημένη και δεν περιορίζεται σε ένα συγκεκριμένο πεδίο εφαρμογών. Με τη βοήθειά τους μπορούμε να δώσουμε λύσεις σε αρκετά σύνθετα προβλήματα και μάλιστα με άνεση και ταχύτητα! Κάτι αντίστοιχο ισχύει και με τα λεγόμενα σενάρια κελύφους (shell scripts). Μπορείτε να τα φαντάζεστε σαν τα batch files (αρχεία BAT) των Windows, μόνο που στην πραγματικότητα τα shell scripts είναι κλάσεις ανώτερα! Τα σενάρια κελύφους δεν περιορίζονται σε μια απλή παράθεση εντολών. Τις περισσότερες φορές περιέχουν κατάλληλες προγραμματιστικές δομές, οι οποίες προσδίδουν την απαιτούμενη εσωτερική λογική αλλά και το στοιχείο της αλληλεπίδρασης με το περιβάλλον. Με άλλα λόγια, τα σενάρια μπορούν να περιλαμβάνουν επαναληπτικές δομές ή δομές ελέγχου και διακλάδωσης, ώστε να προσαρμόζουν τη συμπεριφορά τους ανάλογα με τις παραμέτρους που έχουν δεχτεί κατά την εκτέλεση! Σε αυτό το άρθρο, βέβαια, δεν θα κάνουμε μαθήματα προγραμματισμού. Αντίθετα, θα θεωρήσουμε ότι έχετε μερικές βασικές γνώσεις και θα προχωρήσουμε σε μία γνωριμία με τη γλώσσα του BASH, μέσα από μια σειρά συγκεκριμένων παραδειγμάτων. Πριν ξεκινήσουμε, όμως, πρέπει να αναφέρουμε μερικές λεπτομέρειες. Κατ' αρχάς, όλα τα σκριπτάκια που θα δούμε (όπως κι εκείνα που ελπίζουμε ότι θα αρχίσετε να δημιουργείτε μόνοι σας) πρέπει να γράφονται σε κάποιον διορθωτή απλού κειμένου. Για παράδειγμα, στην κονσόλα μπορούμε να χρησιμοποιούμε το nano, ενώ αν βρισκόμαστε σε παραθυρικό περιβάλλον μπορούμε να χρησιμοποιούμε προγράμματα όπως τα Gedit, Kate και πάει λέγοντας. Αφού γράψουμε κι αποθηκεύσουμε το scriptάκι μας, για να μπορέσουμε να το εκτελέσουμε πρέπει προηγουμένως να τροποποιήσουμε κατάλληλα τα δικαιώματα πρόσβασης σε αυτό, ώστε να γίνει εκτελέσιμο. Κάτι τέτοιο μπορεί να γίνει πανεύκολα με το προγραμματάκι chmod. Για παράδειγμα, αν έχουμε φτιάξει ένα script με το όνομα my_super_script, αρκεί να δώσουμε κάτι σαν αυτό:

```
chmod 744 my_super_script
```

Με αυτό τον τρόπο εμείς θα μπορούμε να το εκτελούμε κανονικά, ενώ οι υπόλοιποι χρήστες θα μπορούν μόνο να το διαβάζουν. Τέλος, σημειώστε ότι για την εκτέλεση ενός σεναρίου πρέπει να δηλώνουμε στο κέλυφος τη θέση του. Έτσι, όταν βρισκόμαστε στον κατάλογο όπου βρίσκεται το σενάριό μας, αρκεί να δίνουμε κάτι τέτοιο:

```
./my_super_script
```

Οι χαρακτήρες ./ συμβολίζουν τον τρέχοντα κατάλογο. Άλλο παράδειγμα: Αν το σκριπτάκι μας είναι αποθηκευμένο στον κατάλογο ~/scripts ενώ εμείς βρισκόμαστε κάπου αλλού, θα πρέπει να δώσουμε κάτι τέτοιο:

```
~/scripts/my_super_script
```

Και μετά από αυτές τις βαρετές εισαγωγές, μπορούμε να περάσουμε στα παραδείγματα που σας υποσχεθήκαμε. Σημειώστε μόνο ότι σε ορισμένες περιπτώσεις δεν θα βρείτε ολόκληρο τον κώδικα των σεναρίων, αλλά μόνο τα αποσπάσματα που θεωρούμε περισσότερο ενδιαφέροντα. Σε αυτές τις περιπτώσεις θα μπορείτε να κατεβάσετε τον κώδικα από τα σχετικά links, στην αρχή της εκάστοτε ενότητας.

Έξυπνη διαγραφή, http://bit.ly/dh016delete

Η διαγραφή ενός αρχείου από το κέλυφος του Linux δεν είναι αντιστρέψιμη! Κάθε φορά που σβήνουμε κάτι από την κονσόλα αυτό διαγράφεται οριστικά και δεν μεταφέρεται σε κάποιον «κάδο ανακύκλωσης». Ακριβώς αυτό το κενό προσπαθεί να καλύψει το σενάριο ονόματι delete. Αφού το κατεβάσετε και φροντίσετε ώστε να γίνει εκτελέσιμο, μπορείτε να το χρησιμοποιήσετε δίνοντας κάτι τέτοιο:

```
./delete όνομα_κάποιου_αρχείου
```

Με αυτόν τον τρόπο, το αρχείο που προσδιορίσατε θα μεταφερθεί σε έναν κρυφό κατάλογο στο home directory, ονόματι .my-trash. Σημειώστε ότι αν ο συγκεκριμένος κατάλογος δεν υπάρχει, το σκριπτάκι μας θα φροντίσει να τον δημιουργήσει. Στην περίπτωση που αργότερα μετανιώσετε για τη διαγραφή του αρχείου, αρκεί να μεταβείτε στον κατάλογο .my_trash και να ρίξετε μια ματιά τριγύρω. Με αυτό το σενάριο θα μπορούσαμε να αντικαταστήσουμε το πρόγραμμα rm, τουλάχιστον για όταν επιτελούμε κρίσιμες εργασίες και χειριζόμαστε πολύτιμα αρχεία. Κάτι τέτοιο θα μπορούσε να γίνει με το κατάλληλο alias. Για παράδειγμα, αν έχουμε αποθηκεύσει το delete στον κατάλογο ~/my_scripts, θα μπορούσαμε να ορίσουμε το ακόλουθο alias:

```
alias del='~/my_scripts/delete'
```

...ή και το ακόμα πιο δραστικό:

```
alias rm='~/my_scripts/delete'
```

Αυτόματο άδειασμα του «κάδου», http://bit.ly/dh016crondel

Το σενάριο delete μπορεί να μας σώσει από μία καταστροφική απροσεξία. Ωστόσο, η συνεχής του χρήση μάλλον θα αυξήσει γρήγορα και υπερβολικά το μέγεθος του καταλόγου ~/.my-trash. Κάπως έτσι, προκύπτει η ανάγκη ενός μηχανισμού ο οποίος θα φροντίζει με κάποιον τρόπο να αδειάζει τον εν λόγω κατάλογο. Το σενάριο που μπορεί να κάνει αυτή τη δουλειά είναι το cron_del και, όπως δηλώνει το όνομά του, προορίζεται για αυτόματη εκτέλεση με τη βοήθεια του crontab. Η αποστολή του είναι να διαγράφει οτιδήποτε μεταφέρθηκε στον κατάλογο .my-trash την προηγούμενη εβδομάδα[1]. Αν υποθέσουμε ότι το έχουμε αποθηκεύσει στον κατάλογο /home/pvar/my_scripts, θα πρέπει να προσθέσουμε στο crontab μία γραμμή σαν την ακόλουθη:

```
0    10    *    *    *    /home/pvar/my_scripts/cron_del
```

Με αυτόν τον τρόπο το cron_del θα εκτελείται κάθε Κυριακή, στις δέκα το πρωί.

Έξυπνη αποσυμπίεση, http://bit.ly/016unpack

Για τον κόσμο του Unix διατίθεται μια πληθώρα προτύπων και αντίστοιχων εργαλείων συμπίεσης ή συσκευασίας αρχείων. Βέβαια, τα επικρατέστερα από όλα είναι τα εξής τρία: bzip2, gzip και tar. Υπάρχει όμως πραγματικός λόγος να θυμόμαστε απ' έξω τη σύνταξη τριών διαφορετικών εργαλείων; Η αλήθεια είναι ότι υπάρχει, αλλά όταν θέλουμε να αποσυμπιέσουμε κάτι στα γρήγορα δεν υπάρχει κανένας λόγος να μπλέκουμε με το συντακτικό

1. Προσοχή: Για το σύστημα η εβδομάδα αλλάζει κάθε Κυριακή. Εξαιτίας της απλότητας του σεναρίου μας, έχετε υπόψη ότι αν σβήσετε κάτι το Σάββατο με το σενάριο delete, την επόμενη μέρα θα διαγραφεί οριστικά από το cron-del!

αυτών των εργαλείων. Γι' αυτό το λόγο δημιουργήσαμε το σενάριο unpack. Όταν το εκτελούμε, ελέγχει αν το αρχείο που δώσαμε σαν παράμετρο είναι συμπιεσμένο. Εφόσον είναι, ανιχνεύει τη μέθοδο συμπίεσης και προχωρά στην αποσυμπίεσή του, χρησιμοποιώντας το κατάλληλο εργαλείο! Έτσι, για να αποσυμπιέσουμε στα γρήγορα ένα πακέτο αρκεί να δώσουμε κάτι τέτοιο:

```
~/my_scripts/unpack ονομα_συμπιεσμένου_αρχείου
```

Και τώρα, μελέτη!

Πλέον, έχετε πάρει μια βασική ιδέα για το τι μπορεί να φτιάξει κανείς στο BASH. Θέλουμε να πιστεύουμε ότι έχετε αρχίσει να ενθουσιάζεστε και να σκεφτόσαστε τι άλλο θα μπορούσατε να φτιάξετε. Ωστόσο, δεν έχουμε πει τίποτα ακόμη για το πώς λειτουργούν τα σκριπτάκια μας κι εμείς δεν τα συνηθίζουμε αυτά — έτσι δεν είναι; Στη συνέχεια, λοιπόν, θα εξετάσουμε ορισμένα σημεία των σεναρίων τα οποία θεωρούμε ότι παρουσιάζουν ιδιαίτερο ενδιαφέρον. Αρχικά, θα μείνουμε σε ένα υψηλό επίπεδο θεώρησης και σε επόμενα παραδείγματα θα εμβαθύνουμε περισσότερο. Ας ξεκινήσουμε με την πρώτη γραμμή των σεναρίων, η οποία είναι ίδια για όλα:

```
#!/bin/bash
```

Αυτή η γραμμή δηλώνει το διερμηνευτή (interpreter) που θα αναλάβει την εκτέλεση του σεναρίου. Το κέλυφος BASH δεν είναι ο μοναδικός υποψήφιος interpreter. Για παράδειγμα, ένα σενάριο κελύφους θα μπορούσε να γραφεί σε Perl ή σε Python. Σε αυτές τις περιπτώσεις η πρώτη γραμμή θα ήταν κάπως έτσι...

```
#!/bin/perl
```

...ή κάπως έτσι:

```
#!/bin/python
```

Ας προχωρήσουμε τώρα λίγο παρακάτω, στο σενάριο delete...

```
if [ -f "$filename" ]; then
   echo "The file exists. That's good!"
else
   echo "The file doesn't exist or isn't regular!"
   exit
fi
```

Όπως αναφέραμε προηγουμένως, η γλώσσα του BASH διαθέτει όλες τις παραδοσιακές δομές ελέγχου που προσφέρουν και οι «κανονικές» γλώσσες προγραμματισμού. Στο συγκεκριμένο απόκομμα βλέπουμε ένα παράδειγμα χρήσης της δομής ελέγχου [if...then...else]. Πρόκειται για την απλούστερη δομή ελέγχου και διακλάδωσης. Σε αυτό το κομμάτι, το script ελέγχει για την ύπαρξη του αρχείου που δώσαμε ως παράμετρο. Αν αποτύχει να εντοπίσει τον αρχείο, ενημερώνει το χρήστη και τερματίζει τη λειτουργία του. Διαφορετικά ενημερώνει και πάλι το χρήστη, αλλά συνεχίζει κανονικά την εκτέλεσή του *αφού* προηγουμένως τερματίσει τη δομή [if...then...else]. Ο τερματισμός αυτής της δομής σηματοδοτείται με το «fi». Ας δούμε τώρα κι ένα απόσπασμα από το σενάριο unpack:

```
case $filetype in
"bzip2compressed" )
   echo "The file is compressed with bzip2. Decompressing..."
   bzip2 -d $filename
;;
 ...
* )
   echo "The file is compressed (or packed) with an unknown tool"
   echo "or it is not compressed at all..."
   echo -n "Filetype: "`file $filename | awk '{ print $2 }'`
   echo " "`file $filename | awk '{ print $3 }'`
;;
esac
```

Η ομοιότητα με άλλες γλώσσες προγραμματισμού επεκτείνεται σε όλες τις παραδοσιακές δομές. Στο προηγούμενο απόσπασμα βλέπουμε ένα παράδειγμα της δομής πολλαπλού ελέγχου και διακλάδωσης, ονόματι case. Στη γλώσσα C η αντίστοιχη δομή ονομάζεται switch, ενώ στην BASIC αποκαλείται select case. Σε αυτή την περιοχή, το σκριπτάκι μας ελέγχει τον τύπο του αρχείου και προχωρά στην αποσυμπίεση με το κατάλληλο εργαλείο. Αν το αρχείο είναι συμπιεσμένο με άγνωστη μέθοδο, το script εμφανίζει τον τύπο του αρχείου κι ενημερώνει το χρήστη για την αποτυχία του. Εδώ αξίζει να σταθούμε για λίγο και να εξετάσουμε μια ενδιαφέρουσα λεπτομέρεια. Παρακάτω φαίνεται η γραμμή που τυπώνει τον τύπο του επιλεγμένου αρχείου:

```
filetype=`file $filename | awk '{ print $2$3}'`
```

Βλέπουμε μια ιδιαίτερα ενδιαφέρουσα δυνατότητα του BASH. Ο λόγος γίνεται για το λεγόμενο command substitution. Παρατηρείστε λίγο τον κώδικα. Το BASH θα εκτελέσει όλα όσα υπάρχουν ανάμεσα στους χαρακτήρες «`» (back-tick). Στη συνέχεια, το αποτέλεσμα από αυτήν την εκτέλεση θα αποδοθεί ως τιμή στη μεταβλητή filetype. Ας δούμε τώρα τι γίνεται ανάμεσα στα back-ticks. Αρχικά εκτελείται το πρόγραμμα file, με παράμετρο το αρχείο που δώσαμε κατά την εκτέλεση του script. Το όνομα αυτού του αρχείου έχει αποδοθεί στη μεταβλητή $filename. Το αποτέλεσμα αυτής της εκτέλεσης θα είναι η εμφάνιση μιας ολόκληρης πρότασης, η οποία θα περιγράφει τον τύπο του συγκεκριμένου αρχείου. Αυτή η πρόταση, με τη βοήθεια της διασωλήνωσης, θα σταλεί ως είσοδος στο πρόγραμμα awk. Με τη βοήθεια του τελευταίου, απομονώνονται και τυπώνονται η δεύτερη και η τρίτη λέξη. Οι συγκεκριμένες λέξεις γνωρίζουμε ότι αποτελούν τον τύπο του αρχείου και ακριβώς αυτές θα αποδοθούν τελικά στη μεταβλητή filetype!

Αυτόματη δημιουργία αναφοράς, http://bit.ly/dh016makereport

Είτε έχουμε πέντε VPS που εξυπηρετούν πενήντα πελάτες το καθένα είτε έναν μικρό server στο σπίτι με μοναδικό πελάτη το PC του διπλανού δωματίου, είναι σίγουρο ότι θα θέλαμε πάντα να γνωρίζουμε την κατάσταση του εκάστοτε συστήματος. Για παράδειγμα, θα είχε ενδιαφέρον αν μπορούσαμε να βλέπουμε εύκολα τα στατιστικά από τη χρήση των αποθηκευτικών μέσων, το σύνολο του τάδε είδους συνδέσεων

που δέχτηκε το μηχάνημα, τη θερμοκρασία των δίσκων και πάει λέγοντας. Στην πράξη, για να βρούμε αυτά τα στοιχεία πρέπει πρώτα να ψάξουμε σε ένα σωρό τοποθεσίες και, φυσικά, να έχουμε συνδεθεί στο μηχάνημα που εξετάζουμε. Δεν θα ήταν καλύτερα αν όλα τα στοιχεία συγκεντρώνονταν αυτόματα; Ε, λοιπόν, ένα μικρό script για το BASH θα μπορούσε πανεύκολα να πετύχει αυτό το στόχο. Στη συνέχεια θα μελετήσουμε μια απλοϊκή εκδοχή ενός τέτοιου script. Μάλιστα, επειδή είμαστε και μερακλήδες, το σενάριό μας προβάλει τα διάφορα στοιχεία σε μια σελίδα HTML. Ας δούμε το εν λόγω script τμήμα προς τμήμα.

Σελίδα αναφοράς

Για τη δημιουργία της σελίδας HTML, το script πρέπει να τυπώνει όσες πληροφορίες συλλέγει ανάμεσα στα κατάλληλα HTML tags. Φυσικά, τα tags και οι πληροφορίες θα μπορούσαν να τυπώνονται με τη χρήση του προγράμματος echo. Ωστόσο, όταν θέλουμε να τυπώσουμε αρκετές γραμμές κειμένου χωρίς να ανησυχούμε για τους ειδικούς χαρακτήρες που ενδέχεται να περιέχουν, είναι καλύτερα να χρησιμοποιούμε το cat. Στις πρώτες γραμμές του script, φαίνεται ένα εξυπηρετικό κολπάκι:

```
#!/bin/bash
cat << REPORTSTART
<html>
<head><title>System Report</title></head>
<body bgcolor="#c0c0c0" text="#000000">
<p><hr size=4></hr></p>
<h1>System Report</h1>
<p><hr size=4></hr></p>
REPORTSTART
```

Με τη σύνταξη που φαίνεται στη δεύτερη γραμμή, το πρόγραμμα cat θα τυπώσει όλα όσα ακολουθούν στο σενάριό μας, μέχρι τη γραμμή που περιέχει τη λέξη «REPORTSTART». Η εν λόγω λέξη δεν αποτελεί κάποια ειδική παράμετρο και στη θέση της θα μπορούσαμε να χρησιμοποιήσουμε οποιαδήποτε άλλη (Σ.τ.Ε. Reykjavik #lemetwra). Έτσι, με τις πρώτες γραμμές κώδικα, το script ορίζει το όνομα της σελίδας και τυπώνει τον τίτλο της.

Βασικά στοιχεία

Σε ποιο μηχάνημα εκτελείται το script και ποιος πυρήνας χρησιμοποιείται; Οι επόμενες γραμμές φροντίζουν για την εκτύπωση αυτών των δύο στοιχείων.

```
echo "<p><b>System name: </b>"
hostname
echo "</p>"
echo "<p><b>Kernel version: </b>"
uname —kernel-release
echo "</p>"
```

Προφανώς, το πρόγραμμα hostname τυπώνει το όνομα του μηχανήματος, ενώ το uname (με τις συγκεκριμένες παραμέτρους) εμφανίζει την έκδοση του πυρήνα που χρησιμοποιείται στο σύστημα. Σε παρόμοιες ενότητες, θα μπορούσαμε να αξιοποιήσουμε πολλά ακόμα προγράμματα του κελύφους. Για παράδειγμα, θα μπορούσαμε να χρησιμοποιήσουμε το uptime, ώστε να εμφανίζεται κι ο συνολικός χρόνος λειτουργίας του συστήματος μετά την τελευταία εκκίνηση.

Επεξεργαστής και μνήμη

Τώρα που είδατε το όνομα του μηχανήματος και την έκδοση του πυρήνα, μπορούμε να περάσουμε σε πληροφορίες που αφορούν στο υλικό του υπολογιστή. Συγκεκριμένα, το script θα τυπώσει το μοντέλο του επεξεργαστή, το μέγεθος της μνήμης RAM και την έκταση του χώρου SWAP.

```
cat << HARDWAREINFO
<p><hr></hr></p>
<p><b>Hardware information: </b></p>
<dl><dd><small><pre>
HARDWAREINFO
echo -e "CPU model: \t \c"
grep 'model name' -m 1 /proc/cpuinfo | awk '{print $5" "$6" "$7" "$8}'
echo -e "Main memory: \t \c"
grep 'MemTotal' /proc/meminfo | awk '{print $2" "$3}'
echo -e "SWAP memory: \t \c"
grep 'SwapTotal' /proc/meminfo | awk '{print $2" "$3}'
echo "</pre></small></dd></dl>"
```

Όπως και στο πρώτο τμήμα που εξετάσαμε, για την εκτύπωση αρκετών διαδοχικών γραμμών HTML χρησιμοποιείται το κολπάκι με το πρόγραμμα cat. Κατ' αυτόν τον τρόπο, τυπώνεται μία επικεφαλίδα για την ενότητα (hardware information) ενώ ταυτόχρονα ξεκινά μία περιοχή (<dl><dd><small>...) στην οποία θα εμφανίζονται τα πάντα με μικρότερο μέγεθος και μετατοπισμένα κατά μία στήλη (ένα TAB) δεξιά. Αμέσως μετά, με τη βοήθεια του grep το σκριπτάκι φιλτράρει τις ζητούμενες πληροφορίες από τα αρχεία /proc/cpuinfo και /proc/meminfo[2]. Σε αυτό το τμήμα, πέρα από τα δύο αρχεία του συστήματος ενδιαφέρον έχει και η χρήση του echo. Με την παράμετρο -e, ενεργοποιούνται οι ειδικές αλληλουχίες χαρακτήρων (sequences) όπως είναι η \t και η \c. Η πρώτη επιβάλει την εκτύπωση ενός χαρακτήρα TAB και η δεύτερη αποτρέπει την εκτύπωση του χαρακτήρα ENTER. Έτσι, οι πληροφορίες που συλλέγει το grep σε κάθε περίπτωση προβάλλονται δίπλα από τα μηνύματα που εμφανίζονται με το echo κι όχι στην από κάτω γραμμή.

Περισσότερα στοιχεία

Στα τμήματα κώδικα που ακολουθούν χρησιμοποιούνται τα τεχνάσματα που συναντήσαμε και προηγουμένως. Για αυτό το λόγο, θα περιοριστούμε σε μία σύντομη περιγραφή των πληροφοριών που συλλέγονται.

2. Τα εν λόγω αρχεία, όπως και πολλά άλλα που βρίσκονται στον κατάλογο /proc, περιέχουν πληροφορίες που αφορούν στα χαρακτηριστικά και στην κατάσταση του υπολογιστή. Αν έχετε την περιέργεια, μπορείτε να δείτε τα περιεχόμενά τους από την κονσόλα, χρησιμοποιώντας το cat ή/και το less.

```
cat << DFINFO
<p><hr></hr></p>
<p><b>Mounted volumes: </b></p>
<dl><dd><small><pre>
DFINFO
df -hT
echo "</pre></small></dd></dl>"
```

Σε αυτή την περιοχή χρησιμοποιείται το df, για την προβολή όλων των προσαρτημένων κατατμήσεων. Η παράμετρος T επιβάλλει την προβολή του είδους κάθε filesystem, ενώ η παράμετρος h καθορίζει σαν μονάδες μέτρησης τα TB, GB, MB και KB, αντί των bytes. Η αναφορά μας θέλουμε να είναι ευανάγνωστη!

```
cat << SYSCRON1
<p><hr></hr></p>
<p><b>Scheduled tasks: </b><tt>(system wide)</tt></p>
<dl><dd> <small><pre>
SYSCRON1
cat /etc/crontab
echo "</pre></small></dd></dl>"
```

Με τις παραπάνω γραμμές εμφανίζονται οι προγραμματισμένες εργασίες που εκτελεί ο cron deamon, για λογαριασμό ολόκληρου του συστήματος. Αυτό επιτυγχάνεται με την εκτύπωση των περιεχομένων του αρχείου /etc/crontab.

```
cat << SSHSTATS
<p><hr></hr></p>
<p><b>SSH statistics: </b></p>
<dl><dd><small><pre>
SSHSTATS
yesterday="`date -d yesterday +%d`"
thismonth="`date -d yesterday +%b`"
cat /var/log/messages | grep sshd | grep "$thismonth $yesterday"
>> /tmp/.my_tmp
tmp1="`cat /tmp/.my_tmp | grep Accepted | wc -l`"
tmp2="`cat /tmp/.my_tmp | grep "Invalid user" | wc -l`"
tmp3="`cat /tmp/.my_tmp | grep "authentication failure" | wc -l`"
echo -e "Valid connections: \t $tmp1"
echo -e "invalid username: \t $tmp2"
echo -e "invalid password: \t $tmp3"
echo "</pre></small></dd></dl>"
```

Σε αυτή την περιοχή το σκριπτάκι αποκτά ιδιαίτερο ενδιαφέρον. Μετά την εκτύπωση των απαραίτητων γραμμών HTML, ορίζονται δύο μεταβλητές. Στη μία αποθηκεύεται η *χθεσινή* ημερομηνία και στην άλλη το όνομα

του μήνα. Οι τιμές των δύο μεταβλητών χρησιμοποιούνται σαν φίλτρο για το grep. Έτσι, καταφέρνουμε να απομονώσουμε όλα τα μηνύματα από το γενικό αρχείο καταγραφής του συστήματος, που τυπώθηκαν την προηγούμενη μέρα. Τα εν λόγω μηνύματα αποθηκεύονται προσωρινά στο αρχείο /tmp/.my_tmp. Στη συνέχεια, με τη βοήθεια του grep και του εργαλείου wc, μετράμε πόσες φορές εμφανίζονται ορισμένα μηνύματα του SSH daemon. Έτσι, το script υπολογίζει και προβάλλει το πλήθος των πετυχημένων συνδέσεων στο σύστημα, καθώς και το πλήθος των αποτυχημένων.

Τερματισμός

Για να ολοκληρωθεί η σελίδα HTML χωρίς σφάλματα, πρέπει να τερματιστούν (να κλείσουν) τα tags που ανοίξαμε στην αρχή του script. Ακριβώς αυτό πετυχαίνουν οι ακόλουθες γραμμές:

```
cat << REPORTEND
<p><hr size=5></hr></p>
</body>
</html>
REPORTEND
```

Το σενάριο που εξετάσαμε συλλέγει μερικές (βασικές) πληροφορίες για το σύστημα και τις χρησιμοποιεί για να δημιουργήσει τον κώδικα μιας σελίδας HTML. Αναρωτιέστε πώς θα πάρουμε αυτόν τον κώδικα σε ένα αρχείο; Αν υποθέσουμε ότι το σενάριο ονομάζεται make_report και ότι βρισκόμαστε στον κατάλογο όπου είναι αποθηκευμένο, αρκεί να δώσουμε κάτι σαν αυτό:

```
./make_report > ~/report_file.html
```

Πλέον, αρκεί να ανοίξουμε το αρχείο report_file.html με κάποιον περιηγητή ιστοσελίδων και να θαυμάσουμε τη σελίδα που δημιούργησε το σενάριό μας :D

Πρόσθετες δυνατότητες

Το σκριπτάκι που είδαμε είναι αρκετά απλό και συλλέγει εξίσου απλές πληροφορίες. Ωστόσο, με λίγη φαντασία και την αντίστοιχη όρεξη για ψάξιμο, μπορούμε να το επεκτείνουμε απεριόριστα. Για παράδειγμα, θα μπορούσαμε να προσθέσουμε μία ενότητα που να προβάλλει τη μέση θερμοκρασία των δίσκων. Αναρωτιέστε πως μπορεί να γίνει κάτι τέτοιο; Τα κομμάτια HTML που θα χρειαστούν είναι παρόμοια με αυτά που είδαμε παραπάνω. Ας δούμε λοιπόν τον υπόλοιπο μηχανισμό, ο οποίος θα υπολογίζει τη μέση θερμοκρασία ενός δίσκου.

Αρχικά, θα πρέπει να εγκαταστήσουμε το εργαλείο hddtemp. Η εγκατάσταση ενός πακέτου πραγματοποιείται με διαφορετικό τρόπο από διανομή σε διανομή, αλλά είμαστε βέβαιοι ότι θα μάθετε εύκολα τι πρέπει να κάνετε στην περίπτωση του δικού σας συστήματος. Στη συνέχεια, πρέπει να προσθέσουμε ένα νέο cronjob, το οποίο θα εκτελείται με αυξημένα δικαιώματα. Η εργασία που θα προσθέσουμε θα εκτελείται ανά μία ώρα και θα αποθηκεύει σε ένα προσωρινό αρχείο την τρέχουσα θερμοκρασία κάποιου δίσκου. όπως καταλαβαίνετε, η λειτουργία του hddtemp απαιτεί τα δικαιώματα του root και ακριβώς γι’ αυτό το αντίστοιχο cronjob πρέπει

να ανήκει στο συγκεκριμένο χρήστη. Για να δημιουργήσουμε ένα cronjob για τον root στο Ubuntu, αρκεί να δώσουμε κάτι τέτοιο:

```
sudo crontab -e
```

Εναλλακτικά, μπορούμε να αποσυνδεθούμε από το δικό μας λογαριασμό, να συνδεθούμε σε εκείνον του root (με sudo su, στο Ubuntu) και να δώσουμε αυτό:

```
crontab -e
```

Τώρα, αν υποθέσουμε ότι θέλουμε να παρακολουθούμε τη θερμοκρασία του δίσκου /dev/sda, αρκεί να δημιουργήσουμε το ακόλουθο cronjob:

```
0    */1     *       *       *       /usr/sbin/hddtemp -n /dev/
sda >> /tmp/.my_disk
```

Έτσι, το σύστημα θα καταγράφει αυτόματα (ανά μία ώρα) τη θερμοκρασία του δίσκου sda στο αρχείο /tmp/.my_disk. Τώρα πρέπει να βρούμε έναν τρόπο για τον υπολογισμό της μέσης θερμοκρασίας. Το script που φαίνεται παρακάτω πετυχαίνει ακριβώς αυτό: Προσθέτει όλες τις τιμές της θερμοκρασίας που βρίσκει στο αρχείο /tmp/.my_disk και στη συνέχεια διαιρεί το άθροισμα με το πλήθος των τιμών. Στο τέλος τυπώνεται το μέγεθος που προκύπτει (ο μέσος όρος) και διαγράφεται το αρχείο με τις παλιές μετρήσεις θερμοκρασίας:

```
#!/bin/bash
for i in `cat /tmp/.my_disk`; do
             tbd=$(($tbd + $i))
             tmp=$(($tmp + 1))
done
tbd=$(($tbd / $tmp))
echo $tbd
rm /tmp/.my_disk
exit
```

Εφόσον έχουμε δημιουργήσει το cronjob που περιγράψαμε νωρίτερα, το script που δημιουργεί τις αναφορές θα μπορεί να καλεί το script που μόλις παραθέσαμε, ώστε να λαμβάνει το μέσο όρο της θερμοκρασίας του δίσκου. Η αναφορά για το σύστημα θα είναι πλέον πληρέστερη!

Επίλογος (λέμε τώρα)

Κάπου εδώ αναγκαζόμαστε και σταματάμε —έστω και προσωρινά— την περιήγησή μας στον κόσμο της γραμμής εντολών και του BASH. Θέλουμε να πιστεύουμε ότι μέσα από αυτή τη μίνι σειρά άρθρων φάνηκε καλύτερα ο αληθινός χαρακτήρας του Linux (καθώς και άλλων UNIX-like συστημάτων). Ελπίζουμε πως όλα όσα συζητήσαμε θα σας κάνουν να εκτιμήσετε περισσότερο τις δυνατότητές του. Στοιχηματίζουμε ότι αρκετοί από εσάς θα το ψάξετε περισσότερο και θα φτάσετε πολύ πιο μακριά. Σε κάθε περίπτωση, πλέον θα είστε σε θέση να παρακολουθείτε ευκολότερα και πάρα πολλά άλλα θέματα που αναπτύσσουμε στο περιοδικό, τα οποία δεν αφορούν πάντα στο Linux ή στο scripting. Κι αν μη τι άλλο, κάτι μας λέει ότι, από εδώ και πέρα, όποτε φορτώνετε το BackTrack θα αισθάνεστε πoooλύ πιο άνετα ;)

Η **διαφήμισή** σας θα μπορούσε να είναι εδώ.

Θα μπορούσε να 'ναι και σε κάποια άλλη σελίδα, δηλαδή.

Αλλά δεν είναι.

Προς το παρόν.

Πάντως μην αναρωτιέστε: Ξέρουμε ότι ξέρετε πως ξέρουμε ότι ξέρετε τι πρέπει να κάνετε.

Μην ντρέπεστε, λοιπόν.

talk2us@deltahacker.gr

Εμείς πάντως θα σας μιλήσουμε. (*)

(*)

Στο μεταξύ συνεχίζουμε, **για δεύτερη συνεχή χρονιά,** με τη φροντίδα του καλύτερου τεχνολογικού περιοδικού γύρω από το ethical **hacking,** τα **δίκτυα** και την **ασφάλεια,** τον **προγραμματισμό** και τα **ηλεκτρονικά.**

Ξέρετε. Για **ανήσυχους, δημιουργικούς ανθρώπους** που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα.